DISEÑO E IMPLEMENTACION DE UNA METAHEURISTICA HIBRIDA BASADA EN RECOCIDO SIMULADO, ALGORITMOS GENETICOS Y TEORIA DE AUTOMATAS PARA LA OPTIMIZACION BI-OBJETIVO DE PROBLEMAS COMBINATORIOS

HENRY DAVID NIETO PARRA

UNIVERSIDAD DEL NORTE

DIVISIÓN DE INGENIERÍAS

MAESTRÍA EN INGENIERIA INDUSTRIAL

BARRANQUILLA

2011

DISEÑO E IMPLEMENTACION DE UNA METAHEURISTICA HIBRIDA BASADA EN RECOCIDO SIMULADO, ALGORITMOS GENETICOS Y TEORIA DE AUTOMATAS PARA LA OPTIMIZACION BI-OBJETIVO DE PROBLEMAS COMBINATORIOS

HENRY DAVID NIETO PARRA

TESIS DE GRADO

Presentado como requisito de grado para optar al título de Magister de Ingeniería Industrial

Director

M.Sc Ing. Elías Niño Ruiz

UNIVERSIDAD DEL NORTE

DIVISIÓN DE INGENIERÍAS

MAESTRÍA EN INGENIERIA INDUSTRIAL

BARRANQUILLA

2011

Nota de aceptación

_____

_____

_____

Elías Niño Ruiz, MSc

Director del proyecto

_____

Ph.D Carmenza Luna

Coordinador Programa Maestría en

Ingeniería Industrial

_____

Corrector

_____

Jurado

_____

Jurado

Viernes 4 de Noviembre de 2011

AGRADECIMIENTOS

A la Universidad del Norte por el apoyo incondicional en mi formación y crecimiento profesional.

A todas las personas que en alguna medida aportaron para que no tuviera dificultades en la elaboración de este trabajo, en especial a la Ingeniera Anangélica Chinchilla Camargo.

Al Ingeniero Elías Niño, por su asesoría y dirección para el logro del objetivo establecido en el desarrollo de este trabajo.

A mis padres, mis suegros, mi esposa y mi hijo quienes fueron el motor activo durante el desarrollo de este trabajo.

RESUMEN

En la actualidad solo un trabajo de investigación se ha dedicado al estudio del espacio factible para problemas combinatorios multi-objetivo basándose en la teoría de Autómatas Finitos Deterministas. La Metaheurística de Intercambio Determinista sobre Autómatas (MIDA), permite modelar y describir de manera eficiente el espacio de soluciones factibles de problemas tipo no polinomial complejo (NP-hard), específicamente al Problema del Agente Viajero (TSP) multi-objetivo.

La tesis de grado presentada a continuación, está basada en MIDA y su principal aporte es el mejoramiento de los resultados obtenidos por ésta al integrar técnicas clásicas de optimización: Recocido Simulado y Algoritmos Genéticos. Al incluir estás dos técnicas, se busca solucionar problemas cada vez más complejos encontrados en diferentes procesos productivos en la industria, con una amplia gama de aplicaciones.

Las métricas utilizadas para comparar la efectividad del algoritmo propuesto, AERSMIDA, fueron escogidas teniendo en cuenta su utilización en otros trabajos especializados en el tema, con los cuales se pudieran realizar comparaciones. Así pues, se tomaron como métricas de comparación la Generación de Vectores No Dominados (GVND), la Distancia Generacional (GD), la Distancia Generacional Inversa (IGD) y el Espaciamiento (S). Los resultados obtenidos se muestran a continuación:

INSTANCIA	ALGORTIMO	GVND	ESPACIMIENTO	GD	IGD
	MIDA	289	0,019288813	14,9446741	2200,00643
KROAB100	AERSMIDA	8479	0,00069021	0,0193521	3,17541823
	MIDA	217	0,03401434	19,1198701	2451,13186
KROAC100	AERSMIDA	7023	0,000794386	0,02794342	5,48380314
	MIDA	281	0,013906191	11,9972745	1807,15845
KROAD100	AERSMIDA	6289	0,00156113	0,03196484	6,42579018
	MIDA	283	0,053267007	13,2513869	2183,78396
KROAE100	AERSMIDA	6440	0,001295607	0,02791734	5,01919415
	MIDA	298	0,01582224	13,7767561	2436,03282
KROBC100	AERSMIDA	6919	0,001454019	0,03398684	7,99217244
	MIDA	241	0,023908639	14,6074	2088,48994
KROBD100	AERSMIDA	5934	0,001386099	0,03708265	8,15998072
	MIDA	280	0,030883013	13,3954499	2424,67171
KROBE100	AERSMIDA	5802	0,002458221	0,03677847	7,84811075
	MIDA	286	0,018445239	11,8873292	1834,38784
KROCD100	AERSMIDA	6301	0,00140016	0,02880871	5,22946373
	MIDA	224	0,02848476	12,6392586	1737,24731
KROCE100	AERSMIDA	4613	0,002570326	0,00176736	0,01440589
	MIDA	228	0,047748166	16,0102025	1720,44948
KRODE100	AERSMIDA	7745	0,001189943	0,02597846	5,22694781

Se busca que la métrica GVND sea un número grande, mientras que GD, IGD, al igual que el espaciamiento, sean números pequeños. Al comparar las métricas obtenidas con AERSMIDA, se pudo constatar que efectivamente el algoritmo desarrollado mejora los resultados arrojados por MIDA. Esto sucede ya que AERSMIDA explora de mejor manera el espacio de soluciones factibles, evitando caer en óptimos locales, gracias a la utilización de las técnicas de Recocido Simulado y Algoritmos Genéticos.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

1. INTRODUCCION

1.1. PLANTEAMIENTO DEL PROBLEMA

En la vida cotidiana encontramos infinidad de problemas combinatorios, tales como la distribución de recursos, optimización de procesos metalurgia[1], procesos de amalgamación en la minería de oro[2], construcción de concentradores eléctricos[3], solo por nombrar unos pocos. En todos estos problemas es necesario optimizar uno o más recursos logrando mejorar las condiciones del proceso y disminuir las pérdidas involucradas en él.

En la industria cada vez se desea optimizar más y más las variables de proceso, por lo que los problema se hacen igualmente más complicados de solucionar, dado que al incrementar el número de elementos también incrementa el espacio de las soluciones factibles de manera prácticamente incalculable, por lo que obtener un óptimo global se vuelve una tarea difícil en un tiempo polinómico.

Entonces, el problema radica en que al aumentar el número de objetivos y restricciones del proceso, así mismo aumentará la complejidad para encontrar un óptimo para el problema planteado, dado el enorme tamaño del espacio de las soluciones factibles.

---

[1]Torralba, Jose. "Optimización de un proceso de metalurgia de polvos. Análisis experimental de los factores influyentes y sus interacciones", Universidad Politécnica de Madrid, Madrid, España.

[2] Pantoja, Freddy. "Optimización del Proceso de la Amalgamación en la Pequeña Minería del Oro: Mejora de su Recuperación y Disminución de las Perdidas de Mercurio", Universidad Autónoma de Madrid, Madrid, España.

[3]Wang, F.K.; Richards, G.W. "Using combinatorial designs to construct partial concentrators", Communications, IEEE Transactions, Vol 39, No 7, pag 1141-1146, Julio, 1991.

Se propone una metaheurística para dar solución a este tipo de problemas, más específicamente el Problema del Agente Viajero (TSP por sus siglas en inglés), el cuál es un caso típico de optimización encontrado en la industria para lo cual se hará uso de las técnicas de recocido simulado, algoritmos genéticos y teoría de autómatas finitos determinista que permita la optimización no solo a TSP sino también, en un futuro, de otros problemas de tipo combinatorios bi-objetivo tales como el ruteo de vehículos, el problema de la mochila y la p – mediana, en adición se compararán los resultados obtenidos con otras técnicas especializadas en el tema.

1.2. OBJETIVOS

1.2.1. Objetivo General

Diseño e implementación de una metaheurística basada en Autómatas Finitos Deterministas, Recocido Simulado y Algoritmos Genéticos para la optimización en dos objetivos de problemas del tipo combinatorio.

1.2.2. Objetivos Específicos

- Diseñar un Autómata Finito Determinista que permita representar el espacio de soluciones factibles para problemas del tipo combinatorio bi-objetivo.

- Diseñar una metaheurística hibrida entre la Metaheurística de Intercambio Determinístico (MIDA), Recocido Simulado y Algoritmos Genéticos para la Optimización Multi-objetivo del tipo combinatorio bi-objetivo.

- Comparar los resultados obtenidos con la técnica propuesta con metaheurística Multi-Objetivos de la literatura especializada.

1.3. CONTRIBUCIONES

En el campo de las ingenierías constantemente nos enfrentamos a solucionar problemas multiobjetivos tales como (programación de producción, scheduling, etc). El resultado de este trabajo es una metaheurística la cual arroja mejores resultados al problema estadístico TPS. Es importante aclarar que los parámetros utilizados por AERSMIDA son los mismos utilizados por MIDA con el objetivo de poder realizar una comparación objetiva de los resultados arrojados por las dos metaheurísticas. El programa trabaja con matrices de pesos y optimiza problemas combinatorios de dos objetivos. Se diseñó un Autómata basado en la Metaheurística de Intercambio Determinista sobre Autómatas (MIDA), la cual permite el modelado del espacio de soluciones factibles de un problema combinatorio multi – objetivo. Debido a esto, se puede definir y describir el espacio de soluciones factibles de un problema combinatorio multi – objetivo evitando así verificar la factibilidad de las soluciones encontradas. El Autómata diseñado evita caer en óptimos locales por medio de las teorías de Recocido Simulado y Algoritmos Genéticos, lo cual hace posible diversificar la población de vectores no dominados que optimizan el problema planteado. Para constatar la efectividad del algoritmo desarrollado, se utilizaron las métricas de comparación Generación de Vectores No Dominados (GVND), Distancia Generacional (GD), Distancia Generacional Inversa (IGD) y Espaciamiento (S), ya que son las mismas utilizadas por MIDA en el análisis de resultados.

1.4. ANTECEDENTES

Los esfuerzos investigativos se han invertido en la inteligencia y métrica aplicadas a técnicas metaheurísticas. El estudio de la representación del espacio factible de soluciones de problemas combinatorios de optimización no ha sido explorado de manera exhaustiva. Uno de los estudios más reciente del tema fue hecho por Niño y Ardila[4] en el año 2009, en donde proponen un Autómata Finito Determinista de Intercambio (AFD-I) que permite representar el espacio de soluciones factibles para un problema combinatorio, lo que ha sido de gran ayuda en la industrial para solucionar cierto tipo de dificultades. Sin embargo, solo soluciona problemas combinatorios mono-objetivo, lo cual no permite resolver problemas más complejos encontrados en la industria.

Dorigo en el año 1996[5], en su tesis doctoral, propuso un algoritmo basado en el comportamiento natural de las hormigas. Esta tesis fue nombrada algoritmo basado en Colonia de Hormigas - ACO por sus siglas en inglés. Hoy en día se pueden encontrar diferentes técnicas metaheurísticas, gracias a las investigaciones hechas por Dorigo y cuya finalidad es aproximar las soluciones a los óptimos del problema. También se encuentra una gran cantidad de algoritmos basados en este sistema, encaminados a problemas multi - objetivo (Multi – Objective Ant Colony Optimization - MOACO).

Iredi en el año 1998[6], propone una técnica para solucionar el problema multi – objetivo del ruteo de vehículos, llamada Hormiga – Bicriterio (BCA). El BCA aplica el concepto de feromona de las hormigas, generando una matriz de ésta por cada objetivo del problema. Es

[4] Niño, Elías; Ardila, Carlos. "Algoritmo Basado en Autómatas Finitos Deterministas para la obtención de óptimos globales en problemas de naturaleza combinatoria", Revista de Ingeniería y Desarrollo, No 25, pag. 100 – 114. ISSN 0122 – 3461, 2009.

[5] Dorigo, Marco; Maniezzo, Vitorio; Colorni, Alberto. "The Ant System: Optimization by a Colony of Cooperating Agents", Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions, Vol 26, No 1, pag. 29-41, Febrero, 1996.

[6] Iredi, Steffen ; Merkle, Daniel; Middendorf, Martin. "Bi-criterion optimization with multi colony ant algorithms", First International Conference on Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science, pag 359–372, 2001.

decir, si el problema tiene n objetivos, se generaran n matrices de feromona. Iredi especifica una heurística propia donde este obliga a las hormigas visitar diferentes regiones del Frente de Pareto.

Mariano y Morales en el año 1999[7] desarrollan un modelo Multi – objetivo llamado Multi – objective Ant – Q ó MOAQ, con la idea de solucionar un problema de las redes de distribución del agua. La finalidad de MOAQ, es que una vez asignadas funciones a una familia de agentes, asignarles a su vez a cada cual uno de los objetivos del problema, ver el comportamiento de las demás funciones cada vez que un agente encuentra una optimización del problema en una función.

Gambardella en el año 1999[8], desarrolla un algoritmo con la finalidad de solucionar el problema de ruteo de vehículos llamado (Multiple ant colony system for vehicle routing problem with time Windows ó MACS-VRPTW). MACS propone una colonia n de hormigas para n funciones objetivo donde cada colonia de hormigas esta encargada de optimizar su función objetivo del problema.

Doerner en el año 2003[9], idea una técnica de optimización de pareto, utilizando la Colonia de Hormigas, esta es llamada (Pareto – Ant Colony Optimization ó PACO). Esta técnica consta de varias matrices de feromona, una para cada objetivo, donde adicionalmente un vector de pesos mezcla proporciones de las matrices de feromona para este determinar la próxima ciudad a visitar por parte de cada hormiga a partir de una ciudad origen con la

[7]Mariano, Carlos; Morales, Eduardo. "A multiple objective Ant-Q algorithm for the design of water distribution irrigation networks", Technical Report HC-9904, Instituto Mexicano de Tecnología del Agua, Mexico, Junio, 1999.

[8]Gambardella, Luca; Taillard, Éric; Agazzi, Giovanni. "MACS-VRPTW: A Multiple Colony System For Vehicle Routing Problems With Time Windows", New Ideas in Optimization, McGraw-Hill, pag.. 73–76, 1999.

[9]Doerner, Karl; Gutjahr, Walter; Hartl, Richard; Strauss, Christine; Stummer, Christian. "Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection", Annals of Operations, pag. 79–99, 2004.

finalidad de comparar las soluciones ya que esto ayuda a explorar nuevas regiones en el Frente de Pareto.

Cardoso en el año 2003[10], aplicando un algoritmo idea la optimización de redes de computación (Multi – Objective Network ACO ó MONA). Donde se puede suponer que cada hormiga es un mensaje que se transmitirá por la red y este mensaje teniendo en cuenta las matrices de feromonas se enruta de acuerdo al número de objetivos del problema.

Doerner en el año 2003[11], desarrolla la Competencia de Hormigas que es un esquema competitivo de Colonias de Hormigas llamado (COMPETants ó CA). Este fue aplicado en problemas con dos objetivos, donde por cada objetivo existe una colonia de hormigas, la colonia que obtenga las mejores soluciones en una iteración, tendrá mayor cantidad de hormigas en la siguiente iteración.

Ideri[12] idea un algoritmo en las Colonias de Hormigas basado en Bicriterio (BCA) y técnicas evolutivas llamado Bicriterio MC (BCM). Esta una generalización del BCA aplicado a problemas multi – objetivo utilizando cruzamiento de soluciones para evitar óptimos locales. La tabla 1 ilustra la taxonomía de los algoritmos previamente mencionados.

Tabla 1. Taxonomía para los MOACOS. Clasificación de los problemas basados en colonias de hormigas para la optimización multi – objetivo de problemas combinatorios.

	Una sola matriz heurística	**Múltiple matrices heurística**
Una sola matriz de feromona.	MOACOM	MOAQ - MACS
Múltiples matrices de feromona.	PACO	BicriterionAnt (BA)
		UnsortBicriterion (USBC)
	MONA	CompentAnts (CA)
		MACS – VRPTW

[10]Cardoso, Pedro; Jesús, Mario; Marquez, Alberto. "MONACO- Multi-Objective Network Optimization based on an ACO", Proc. X Encuentros de Geometría Computacional, Sevilla, España, Junio 16–17, 2003.

[11] Doerner, Karl; Hartl, Richard; Reimann, Marc. "Are COMPETants more competent for problem solving?—The Case of Full Truckload Transportation", Central European Journal of Operations Research, pag. 115–141, 2003.

[12] Ibid 6.

Técnicas modernas como SPEA2[13] y NSGA – II permiten la optimización de problemas combinatorios mediante el manejo de frentes elitistas con el objetivo de converger rápidamente a soluciones óptimas. Sin embargo, una rápida convergencia puede culminar con la obtención de óptimos locales.

Por otra parte, técnicas clásicas como las estrategias de búsqueda local son utilizadas para la optimización multi – objetivo actual. Una de las mas poderosas técnicas es la de Recocido Simulado Multi – Objetivo (MOSA – "*Multi – objective simulated annealing*"). Los MOSA son técnicas fiables porque evitan en la medida de lo posible óptimos locales. Serafini (2008)[14] diseño una aproximación a MOSA. Serafinis MOSA (SMOSA) es una estrategia que optimiza problemas combinatorios asignando un peso a cada una de las funciones en cada iteración. La figura 1 ilustra la dirección aleatoria de búsqueda de SMOSA. Ulungu (1999)[15] propone la optimización de un problema combinatorio multi – objetivo mediante la optimización de cada objetivo por medio de una corrida de un SA. Para mantener la diversidad de las soluciones no dominadas, UMOSA trabaja con un vector de pesos asignado de manera balanceada a cada uno de los objetivos. La búsqueda de óptimos de UMOSA puede apreciarse en la figura 2.

[13] Tse Guan Tan; Hui Keng Lau; Teo, J. "Cooperative coevolution for pareto multiobjective optimization: An empirical study using SPEA2", TENCON 2007 - 2007 IEEE Region 10 Conference , Octubre 30 a Noviembre 2, 2007.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4429134&isnumber=4428770

[14] Bandyopadhyay, S.; Saha, S.; Maulik, U.; Deb, K., "A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA," Evolutionary Computation, IEEE Transactions, Vol 12, No 3, pag. 269-283, June 2008.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4358775&isnumber=4531586

[15] Ulungu, E.; Teghem, J.; Fortemps, P.; Tuyttens, D. "MOSA method: A tool for solving multiobjective combinatorial optimization problems," Journal of Multi-Criteria Decision Analysis, vol. 8, no. 4, pp. 221–236, 1999.

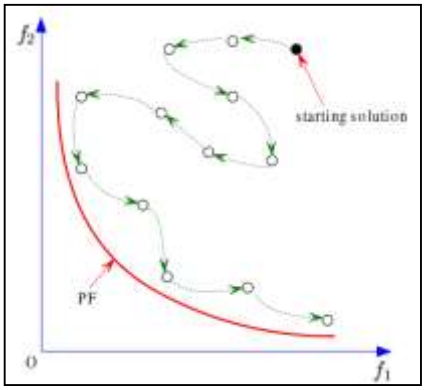

Figura 1. Búsqueda aleatoria de soluciones por parte de SMOSA.

Czyzak (1998) [16] propone un algoritmo de recocido simulado llamado CMOSA. CMOSA es una técnica de SA que trabaja con direcciones adaptativas de acuerdo a los valores de las funciones objetivos. El comportamiento de CMOSA puede apreciarse en la figura 3

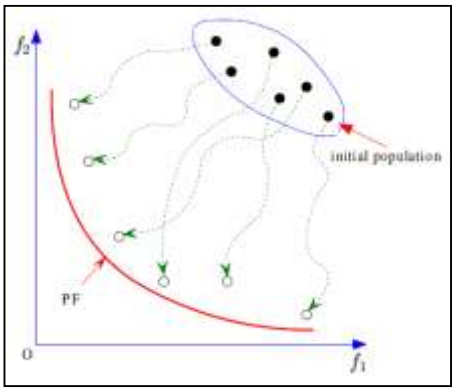

Figura 2. Búsqueda ajustada de soluciones por parte de UMOSA.

[16]Czyzak, P.; Jaszkiewicz, A. "Pareto simulated annealing - a metaheuristic technique for multiobjective combinatorial optimization," Journal of Multi-Criteria Decision Analysis,Vol 7, No 1, pag. 34–37, Diciembre, 1998.

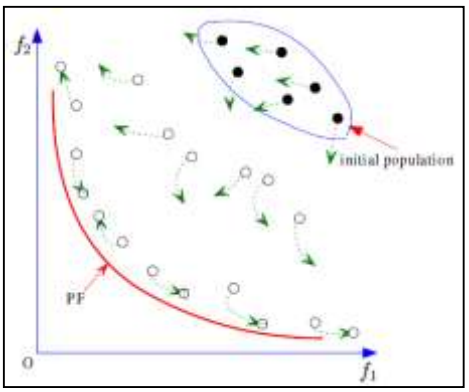

Figura 3. Búsqueda adaptativa de soluciones por parte de CMOSA.

Recientemente Li y Landa – Silva (2008) diseñaron una estrategia basada en Algoritmos Evolutivos y MOSA. Un Algoritmo de Recocido Simulado Evolutivo llamado EMOSA. EMOSA aprovecha las ventajas de ajuste de UMOSA y las cualidades adaptativas de CMOSA. La dirección de búsqueda de EMOSA puede apreciarse en la figura 4.

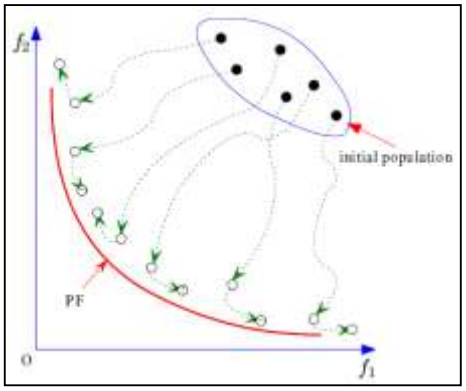

Figura 4. Búsqueda aleatoria de soluciones por parte de SMOSA.

Aún cuando existen muchas técnicas asociadas a la solución de problemas combinatorios multi – objetivo, predominan las falencias en los tiempos de respuestas, como por ejemplo los algoritmos MOSA. Adicionalmente, la sensibilidad de las variables de salida con respecto a los parámetros del algoritmo, son un factor determinante en la consecución de las soluciones óptimas. Muchas veces establecer los valores a los parámetros de un algoritmo (como por ejemplo ACO) exige un análisis fuerte del problema.

Otro aporte importante al problema del agente viajero multi-objetivo fue hecho por Borgulya en el año 2007[17]. También desarrolló otro algoritmo del tipo evolutivo, basado en la asignación cuadrática bi-objetivo a partir del mismo problema (TSP)[18], que utiliza el método de memoria colectiva o virtual perdedor, que permite trabajar con una mayor cantidad de valores discretos. El algoritmo fue llamado MOSCA2b y a continuación se muestras una comparación con el algoritmo llamado genetic local search for multiple objective combinatorial optimización (MOGLS) tomando como medida de rendimiento el indicador binario ε- indicador y como instancias KROAB50, KROAB100, KROAD50, KROAD100 para dos objetivos y KROABC100, KROABD100 para tres objetivos, ε1 da Iε (B, A), ε2 da Iε (A, B) donde A es el resultados de MOSCA2b y B es el resultado de MOGLS, no hace referencia al número de objetivos utilizados, los resultados son mostrado en la tabla 2. Las instancias KROAB50, KROAB100, KROAD50, KROAD100, KROABC100 y KROABD100, se encuentran en la librería TSPLIB[19] creada y administrada por el Grupo de Investigación de Optimización Discreta de la Universidad de Heidelberg en Alemania, la cual contiene más de 200 instancias para el Problema del Agente Viajero, y se ha convertido en instancias de pruebas estándar para diversos estudios especializados en el tema.

Tabla 2. Los mejores resultados no dominados de MOSCA2b.[22]

No	instancias	ε1	ε2	Ndn
2	KROAB50	1.082	1	253
	KROAD50	1.002	0.999	213
	KROAB100	0.991	1.019	319
	KROAB100	0.974	0.997	312
3	KROABC100	0.979	1.025	2565
	KROABD100	0.977	0.998	2515

[17] Borbulya, István. "An EC-Memory based Method for the Multi-Objective TSP", Proceedings of the 9th annual conference on Genetic and evolutionary computation, New York, 2007.

[18] Borgulya, István. "An Evolutionary Algorithm for the bi-objective QAP", Computational Intelligence, Theory and Applications Advances, springer series, part 22, pag. 577-586, 2006.

[19] TSPLIB. http://comopt.ifi.uni-heidelberg.de/index.html

Se puede observar que los resultados obtenidos por el algoritmo de MOSCA2b son muy similares a los del algoritmo de MOGLS.

Niño (2010)[20] en su tesis de magister, plantea una nueva metaheurística basadas en autómatas finitos determinísticos para la optimización multi-objetivo de problemas combinatorios, como respuesta a la problemática de hallar soluciones óptimas en un espacio de soluciones propuesta por un modelo con AFDM de un problema combinatorio, debido a que la cota de crecimiento del número de estados de Q es exponencial respecto al número de variables de decisión o entrada. El autor plantea una estrategia que inicialmente evita en lo posible caer en óptimos locales, para ello siempre se busca aproximar el conjunto de estados Q_φ que contengan las soluciones óptimas del total de estados Q a partir de Q_0.

La metaheurística MIDA fue validada con instancias del Problema del Agente Viajero (TSP por sus siglas en inglés – "*Traveling Salesman Problem*"). Comparando los resultados de otras técnicas reconocidas como lo son Colonia de Hormigas – Bicriterio (BCA), Bicriterio MC (BCM), Colonia de Hormigas (CA), Multiple Ant Colony System (MACS), Multi – objective Ant – Q (MOAQ), Multi – Objective Network ACO (MONA), Non Dominated Sorting Genetic Algorithm NSGA – II, Pareto – Ant Colony Optimization (PACO), SPEA2 y Unsort Bicriterion (USBC), obteniendo los siguientes resultados para las métricas Generación de Vectores No Dominados (GVND), Espaciamiento, Distacia Generacional (GD) y Distancia Generacional Inversa (IGD) en cada una de las instancias:

[20] Niño, Elías; Ardila, Carlos; Jabba, Daladier; Donoso, Yesid. "A novel Algorithm based on Deterministic Finite Automaton for solving the mono-objective Symmetric Traveling Salesman Problem". International Journal of Artificial Intelligence, North America, 2010.

Tabla 3. Comparación de los resultados obtenidos por MIDA, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA – II, PACO, SPEA2 y USBC utilizando métricas para la instancia KROAB50

	GVND	ESPACIAMIENTO	GD	IGD
BCA	77	0.1150	13.2641	2173.1927
BCM	70	0.1634	30.5382	9520.0810
CA	49	0.0969	44.2144	9778.6389
MIDA	**7302**	**0.0005**	**0.0322**	**115.1132**
MACS	117	0.1019	6.5284	1215.4514
MOAQ	55	0.1362	27.5772	4792.7332
MONA	45	0.1398	54.3892	12479.8477
NSGA2	64	0.2106	28.4399	6901.9911
PACO	38	0.3266	63.0131	11945.0227
SPEA2	40	0.2946	58.7052	11487.6566
USBC	123	0.0401	6.3924	1287.9326

Tabla 4. Comparación de los resultados obtenidos por MIDA, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA – II, PACO, SPEA2 y USBC utilizando métricas para la instancia KROAB100

	GVND	ESPACIAMIENTO	GD	IGD
BCA	80	0.0462	56.6924	10786.4786
BCM	80	0.1907	77.2998	20053.3008
CA	67	0.0588	143.8240	48692.4501
MIDA	**8013**	**0.0012**	**0.0041**	**0.5795**
MACS	129	0.0452	27.1489	6431.7973
MOAQ	74	0.0840	120.1045	41422.0084
MONA	27	0.2460	280.7074	30122.0496
NSGA2	45	0.1378	247.1434	64859.3292
PACO	40	0.1680	187.2472	29417.1143
SPEA2	48	0.2007	228.3903	63021.2864
USBC	184	0.0302	16.3255	4731.7026

Como puede apreciarse en la tabla 4, en las pruebas realizadas, MIDA tuvo mejor comportamiento que el resto de sus comparadas. Vale la pena mencionar, como se observa en la tabla 3, que en la instancia KROBC100, MIDA dominó por completo a todas las técnicas, ya que su respectivo IGD y GD es igual a 0 lo cual implica que los puntos del Frente de Pareto Ideal (FPI) son exactamente iguales a los del Frente de Pareto (FP) obtenido por MIDA.

Tabla 5. Comparación de los resultados obtenidos por MIDA, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA – II, PACO, SPEA2 y USBC utilizando métricas para la instancia KROBC100

	GVND	ESPACIAMIENTO	GD	IGD
BCA	94	0.0358	52.4217	13261.3962
BCM	81	0.0510	65.0868	15179.8542
CA	57	0.0539	162.0395	46591.1182
MIDA	**5806**	**0.0020**	**0.0000**	**0.0000**
MACS	118	0.0841	34.3191	8956.6726
MOAQ	72	0.1238	111.6502	35293.5484
MONA	22	0.3837	342.7103	31046.4039
NSGA2	47	0.2000	232.6861	65320.3249
PACO	36	0.2220	203.6702	29361.0403
SPEA2	40	0.2047	267.1160	62349.2827
USBC	187	0.0322	16.8676	5433.7462

Tabla 6. Promedios de los resultados obtenidos por MIDA, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA – II, PACO, SPEA2 y USBC utilizando métricas.

	GVND	ESPACIAMIENTO	GD	IGD
BCA	83	0.0656	40.7928	8740.3558
BCM	77	0.1350	57.6416	14917.7453
CA	57	0.0698	116.6926	35020.7357
MIDA	**7040**	**0.0013**	**0.0121**	**38.5642**
MACS	121	0.0771	22.6654	5534.6404
MOAQ	67	0.1147	86.4439	27169.4300
MONA	31	0.2565	225.9356	24549.4337
NSGA2	52	0.1828	169.4231	45693.8817
PACO	38	0.2389	151.3101	23574.3924
SPEA2	42	0.2333	184.7372	45619.4086
USBC	164	0.0342	13.1951	3817.7938

2. MARCO TEORICO

Un aspecto diario de la ingeniería es el de enfrentarse a problemas cada vez más complejos que implican ser resueltos o expresados como problemas de optimización. Algunos de estos problemas son solucionados con técnicas tradicionales de la programación lineal y otros son de naturaleza no lineal. Dentro de todo este conjunto de problemas, existe un grupo el cual para altos volúmenes de información su solución es difícil de encontrar en tiempo polinomial. Estos tipos de problemas se denominan No Polinomiales (NP). Aún más interesante es el factor de optimizar a la vez más de un objetivo, como suele suceder en la realidad. Existen diversas técnicas que permiten la optimización de problemas de ingeniería, entre las cuales sobresalen las técnicas metaheurísticas. Las técnicas metaheurísticas brindan apoyo a la solución de problemas diarios de la ingeniería aunque las soluciones encontradas mediante el uso de estas son sensibles a los parámetros de entrada. El presente capitulo tiene como objetivo mostrar las definiciones relevantes en la construcción y definición de la teoría de Autómatas Finitos Deterministas Multiobjetivo y la construcción de la Metaheurística MIDA.

2.1. OPTIMIZACIÓN

Cuando en la industria se presenta un problema, lo primero que hacemos como ingenieros es buscar solucionarlo de la manera más sencilla posible. Luego de ello, el paso lógico a seguir es mejorarlo de tal forma que pueda obtener una solución del problema, pero a la vez optimizarlo.

Para poder optimizar un proceso debe conocerse plenamente. Es decir, manejar todas las variables que están involucradas, los factores que puedan perturbar el proceso, etc. La optimización no solo nos sirve para que el proceso sea más rápido, sino también para mejorar el uso de otras variables de interés, como el personal o el material necesario para tal proceso.

Un problema de optimización está conformado por:

- **Variables de optimización:** Las variables de optimización son los valores que se modifican para resolver el problema[21].

- **Función objetivo:** Un problema de optimización puede tener una o varias funciones objetivo. Generalmente, estas funciones se expresan en términos de las variables de decisión y es la función que se desea optimizar, por lo cual es aquella en la que se evalúan las posibles soluciones. Cuando se tiene una sola función, se considera que es un problema de optimización mono-objetivo y cuando se evalúan más de dos funciones objetivo se habla de un problema de optimización multi-objetivo[22].

- **Restricciones:** Las restricciones se expresan en ecuaciones de igualdad o desigualdad. Para que una solución del problema se considera factible, se deben cumplir con todas las restricciones planteadas en él. En caso de que no tenga ninguna restricción, todas las soluciones de la función objetivo serán válidas para resolver el problema[23].

[21] Cagnina, Leticia. "Optimización Mono y Multiobjetivo a través de una Heurística de Inteligencia Colectiva", Facultad de Ciencias Físico Matemáticas y Naturales, Universidad Nacional de San Luis, Argentina, 2010.

[22] Ibid 21

[23] Ibid 21

Existe la posibilidad de que encontramos varias soluciones para optimizar un mismo problema, es decir, que encontramos varios óptimos que cumplen con todas las restricciones del proceso. Pero la solución que realmente interesa es la solución óptima global, la cual es aquella solución que es la mejor dentro de todas las soluciones factibles. También existe otro grupo de soluciones llamadas soluciones no factibles, las cuales son todas las soluciones que no logran cumplir con todas las restricciones del problema.

2.1.1. OPTIMIZACION MONO-OBJETIVO

Los problemas de optimización son comúnmente representados en un esquema mono-objetivo. En otras palabras, el proceso consiste en optimizar una sola función objetivo teniendo en cuenta una serie de restricciones que están basadas en restricciones del mundo real[24]. Un problema de optimización mono-objetivo es expresado de la siguiente manera:

$$Optimizar\ [maximizar/minimizar]\ f(X) \quad \textbf{(1)}$$

Sujeto a:

$$H(X) = 0 \quad \textbf{(2)}$$

$$G(X) \leq 0 \quad \textbf{(3)}$$

[24] Xin Sui; Ho-Fung Leung, "An Adaptive Bidding Strategy in Multi-round Combinatorial Auctions for Resource Allocation," Tools with Artificial Intelligence, 2008. ICTAI '08. 20th IEEE International Conference, Vol 2, pag. 423-430, 3-5 Noviembre, 2008.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4669804&isnumber=4669739

En este caso, la función a optimizar (maximizar ó minimizar) es $f(X)$, donde el vector X es el conjunto de variables independientes. Las funciones $H(X)$ y $G(X)$ son las restricciones del modelo.

Para este problema existen tres conjuntos soluciones: el **conjunto universal**, el cual está compuesto por todos los posibles valores de X sean factibles ó no. El **conjunto de soluciones factibles**, el cual está compuesto por todos los valores de X que cumplen con las restricciones $H(X)$ y $G(X)$. Por último, el **conjunto de soluciones optimas**, el cual está compuesto por todos los valores de X que, en adición de ser factibles, alcanza el valor óptimo (mínimo ó máximo) de la función $f(X)$, sea en un intervalo específico $[a, b]$ ó en un contexto global $(-\infty, \infty)$. Los tres conjuntos mencionados se pueden apreciar en la figura 5.

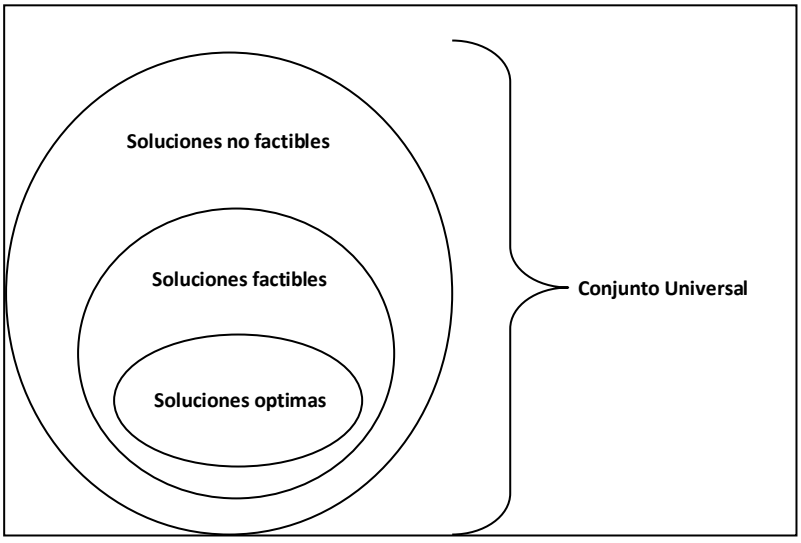

Figura 5. Conjuntos soluciones para un problema mono-objetivo. Los tres posibles conjuntos soluciones en un problema mono-objetivo, lo ideal es alcanzar el conjunto de soluciones óptimas.

2.1.2. OPTIMIZACION MULTI-OBJETIVO

Si es cierto que los esquemas mono-objetivo ayudan a la solución de problemas, en la vida real cuando se desea solucionar un problema muchas veces interesa optimizar más de una función[25]. Debido a que existen varias funciones a optimizar, hablamos de un problema de optimización multi-objetivo. Un problema de optimización multi-objetivo es expresado de la siguiente manera:

$$Optimizar\ [maximizar/minimizar]\ F(X) = \{f_1(X), f_2(X), \dots, f_n(X)\} \quad \textbf{(4)}$$

Sujeto a:

$$H(X) = 0 \quad \textbf{(5)}$$

$$G(X) \leq 0 \quad \textbf{(6)}$$

En este caso, las funciones a optimizar (cada una puede ser maximizada ó minimizada) son el conjunto de funciones $F(X)$, donde X es el conjunto de variables independientes. Las funciones $H(X)$ y $G(X)$ son las restricciones del modelo. Debido a que existe más de una función a optimizar, es posible que la mejora de una perjudique al resto de funciones.

2.1.2.1. FRENTE DE PARETO (FP)

La optimización de un problema multi-objetivo es completamente diferente a la de un problema mono-objetivo. En un problema mono-objetivo se busca la mejor solución de todas y bajo este esquema solo existe un punto de comparación por el cual se decide si una

[25]Groselj, B.; Malluhi, Q.M., "Combinatorial optimization of distributed queries," Knowledge and Data Engineering, IEEE Transactions, Vol 7, No 6, pag. 915-927, Diciembre, 1995.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=476497&isnumber=10156

solución es mejor que otra[26], por ejemplo, supongamos el conjunto de soluciones de la figura 6, cada solución puede ser contrastada con el resto de soluciones mediante un único valor. En el caso de que las soluciones correspondieran a un problema de minimización, la mejor solución sería f_3 por ser el menor del conjunto. Contrario a esto, si el problema fuese de maximización, la mejor solución sería f_2 por ser el mayor del conjunto. En un problema multi-objetivo no existe un único punto de referencia por el cual se puede decir que una solución es mejor que otra, por ejemplo, supongamos las soluciones de la figura 7 para un problema con dos objetivos. Si todos los objetivos se minimizarán, no se puede determinar entre las dos primeras soluciones cual es mejor ya que $f_{11} > f_{21}$ y $f_{12} < f_{22}$, sin embargo, las dos primeras soluciones son mejores que la tercera, ya que para este caso en particular ambas funciones son menores. En este caso se dice que la función 3 es dominada por la solución 1 y la solución 2.

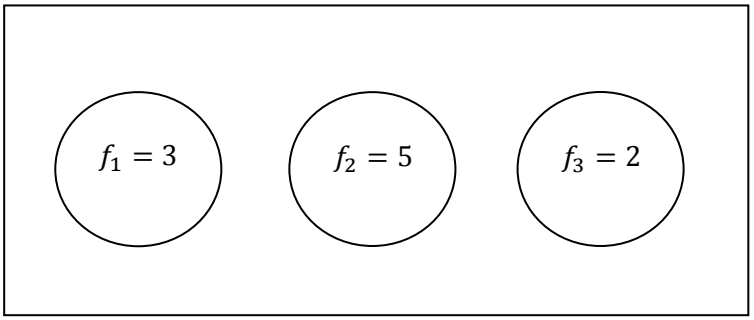

Figura 6. Conjunto de soluciones mono-objetivo. Tres soluciones para un problema particular mono-objetivo. Si se tratase de un problema de maximización la mejor solución sería f_2. En el caso contrario, si el problema fuese de minimización, la mejor solución sería f_3.

[26]Gen, Mitsuo; Cheng, Runwei. "Genetic Algorithms & Engineering Optimization", Wiley-Interscience Publications, pag. 97-106, New York, Estados Unidos, 2000.

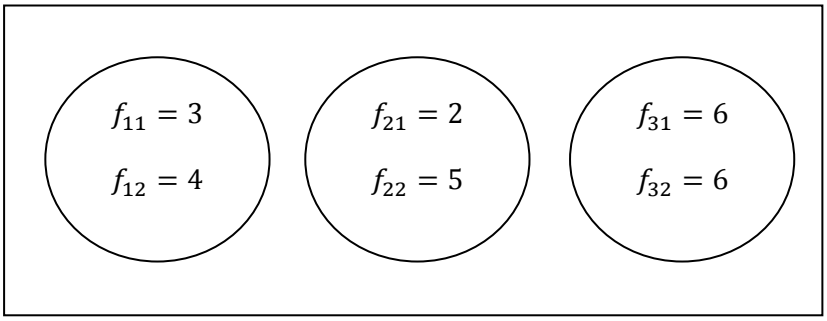

Figura 7. Conjunto de soluciones multi-objetivo. Tres soluciones para un problema particular con dos objetivos.

Formalmente, se dice que una solución a **domina a** una solución b si y solo si:

$$(\forall f_{ai} \in a)(\forall f_{bi} \in b)(f_{ai}\ es\ mejor\ que\ f_{bi}) \quad \textbf{(7)}$$

Formalmente, se dice que una solución a **no domina a** una solución b si y solo si:

$$(\forall f_{ai} \in a)(\exists f_{bi} \in b)(f_{ai}\ no\ es\ mejor\ que\ f_{bi}) \quad \textbf{(8)}$$

El conjunto solución de un problema multi-objetivo se conoce con el nombre de **soluciones no dominadas**[27], en el caso particular de dos y tres objetivos, las soluciones no dominadas forman una gráfica conocida con el nombre de **Frente de Pareto (FP)[28],** un ejemplo de FP puede ser apreciado en la figura 8.

[27]Donoso, Yezid; Fabregat, Ramón. "Multi-Objective Optimization in Computer Networks Using Metaheuristics", Aurbach Publications, pag. 1 – 3, New York, Estados Unidos, 2007.

[28] Ibid 26.

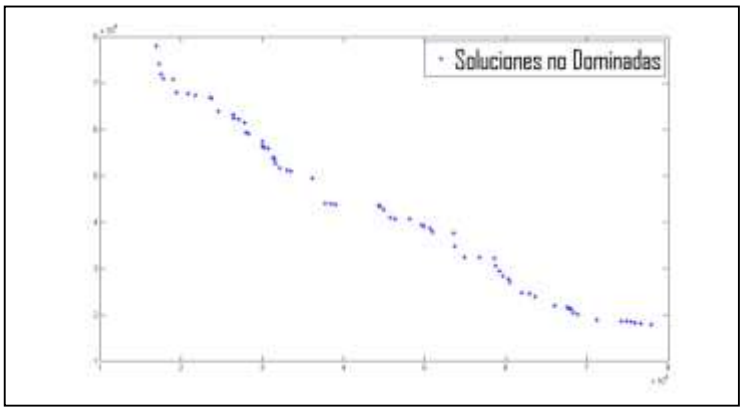

Figura 8. FP Bi-Objetivo. Conjunto de soluciones no dominadas para un problema en particular con dos objetivos.

2.1.2.1.1. METRICAS

Las métricas son utilizadas para medir la calidad de soluciones no dominadas, estas pueden ser singulares ó en contraste con otro conjunto solución. La calidad de las soluciones dependerá del tipo de métrica utilizado. A continuación se mostraran algunas métricas más importante y relevantes.

2.1.2.1.1.1. Generación de Vectores No Dominados (GVND)[29]

La métrica GVND, índica el número de elementos no dominados generados por el algoritmo, es decir nos muestra cuántos elementos hay en el FP, esto es:

$$GVND = |FP_{real}| \quad \textbf{(9)}$$

[29]Tovar, Luis; Coronell, Margarita; Donoso Yezid. "Optimización multiobjetivo en redes ópticas con transmisión Multicast, utilizando algoritmos evolutivos y lógica difusa". Ingeniería & Desarrollo. Número 21. Enero-Junio 2007.

2.1.2.1.1.2. Distancia Generacional (GD)[30]

La métrica GD, nos índica qué tan lejos está el FP Aproximado ($\boldsymbol{FP_{aprox}}$) del FP Real ($\boldsymbol{FP_{real}}$), esto es:

$$DG = \frac{\sqrt{\sum_{i=1}^{|FP_{aprox}|} d_i}}{|FP_{aprox}|} \quad \textbf{(10)}$$

Donde d_i es la mínima distancia Euclidiana entre el i-ésimo elemento del FP aproximado con los elementos del FP real.

2.1.2.1.1.3. Distancia Generacional Inversa (IGD)[31]

La métrica DG, nos índica qué tan lejos está el FP $\boldsymbol{A}$ de un FP de Referencia $\boldsymbol{A}^*$, esto es:

$$IGD(A, A^*) = \frac{1}{|A^*|}\sum_{i=1}^{|A|} d_i \quad \textbf{(11)}$$

Donde d_i es la mínima distancia Euclidiana entre el i-ésimo elemento del FP $\boldsymbol{A}$ con los elementos del FP de Referencia $\boldsymbol{A}^*$.

---

[30] Lim, A.; Fan Wang. "Multi-depot vehicle routing problem: a one-stage approach", Automation Science and Engineering, IEEE Transactions, Vol 2, No 4, pag. 397-402, Octubre, 2005.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1514459&isnumber=32436

[31] Zitzler, E.; Thiele, L.; Laumanns, M.; Fonseca, C.: da Fonseca, V. "Performance assessment of multiobjective optimizers: an analysis and review." IEEE Transactions on Evolutionary Computation, Vol 7, No 2, pag. 117–132, 2003.

2.1.2.1.1.4. Espaciamiento (S).[32]

La métrica S, nos índica qué tan bien están distribuidas las soluciones sobre el FP. Matemáticamente se define como:

$$S = \sqrt[2]{\left(\frac{\sum_{i=1}^{|FP_{conocido}|}(\bar{d} - d_i)}{|FP_{conocido}|}\right)} \quad \textbf{(12)}$$

Donde $d_i = min_j\left(\sum_{k=1}^{m}\left|f_k^i(x_i) - f_k^j(x_j)\right|\right)$ con $i \neq j$ y $1 \leq i, j \leq |FP_{conocido}|$ y $\bar{d}$ es el promedio de todos los d_i.

Para esta métrica un valor de 0 significa que todos lo elementos de $FP_{conocido}$ están equidistantemente espaciados. Por lo tanto, entre menor sea el valor de S, mejor es la distribución de los elementos del $FP_{conocido}$.

2.1.2.2. TECNICAS CLASICAS

Las técnicas clásicas de optimización multi-objetivo funcionan aproximando las soluciones del problema planteado mediante la optimización de un problema mono-objetivo[33]. A continuación se mencionan dos de las más conocidas:

[32]Ibid 29.

[33]Ibid 27.

2.1.2.2.1. Suma con Pesos

Este método consiste en crear un modelo mono-objetivo mediante la ponderación de pesos en las n funciones objetivos del problema. A través de la suma con pesos, la expresión 4 puede ser redefinida como:

$$Optimizar\ [maximizar/minimizar]\ F'(X) = \sum_{i=1}^{n} \alpha_i \times f_i \quad \textbf{(13)}$$

Sujeto a:

$$H(X) = 0 \quad \textbf{(14)}$$

$$G(X) \leq 0 \quad \textbf{(15)}$$

$$\sum_{i=1}^{n} \alpha_i = 1, \qquad 0 \leq \alpha_i \leq 1 \quad \textbf{(16)}$$

Como puede apreciarse en 9, $F'(X)$ es una combinación lineal de $F(X)$ debido a la restricción 12.

2.1.2.2.1. ε – Restricción

Esta técnica consiste en crear un modelo mono-objetivo optimizando solo una función de 4 y convirtiendo el resto de funciones en restricciones del problema. A través de ε – Restricción la expresión 4 se convierte en:

$$Optimizar\ [maximizar/minimizar]\ f_i(X) \quad \textbf{(17)}$$

Sujeto a:

$$H(X) = 0 \quad \textbf{(18)}$$

$$G(X) \leq 0 \quad \textbf{(19)}$$

$$f_k(X) \leq \varepsilon_k\ \ , k = 1, \dots, n\ y\ i \neq k \quad \textbf{(20)}$$

En este caso la función $f_i(X)$ es la única a ser optimizada. El resto de funciones objetivos se convierten en restricciones del problema. Para cada $f_k(X)$ existe un valor ε_k que la acota. Cuando es alterado algún ε_k se obtienen nuevas soluciones para 13.

2.2. OPTIMIZACION COMBINATORIA

La optimización combinatoria es un campo de las matemáticas aplicadas, que combina técnicas de combinatoria, programación lineal y la teoría de algoritmos para resolver

problemas sobre estructuras discretas[34]. En esta área se estudian nuevas técnicas que permitan resolver problemas cuya naturaleza es No Polinomial en donde no es posible encontrar soluciones satisfactorias al problema

Los problemas de decisión hacen parte de esta rama de la optimización. Uno de los problemas más representativos en este género es el Problema del Agente Viajero (Traveling Salesman Problem). Este problema consiste en que partiendo de una ciudad de un conjunto de ciudades, se vuelva al origen pasando exactamente una vez por cada una de las ciudades con el menor costo posible. El modelo matemático que representa el problema es el siguiente:

$$optimizar[minimizar|maximizar]\, Z(x) = \sum_{(i,j)\in A} c_{ij}\, x_{ij} \qquad \textbf{(21)}$$

Sujeto a:

$$\sum_{\{i:(i,j)\in A\}} x_{ij} = 1 \;\forall\; j \;\in V \qquad \textbf{(22)}$$

$$\sum_{\{j:(i,j)\in A\}} x_{ij} = 1 \;\forall\; i \;\in V \qquad \textbf{(23)}$$

[34] Hincapie, Ricardo; Rios, Carlos; Gallego, Ramón. "Técnicas Heurísticas aplicadas al problema del cartero viajante (TSP)", Sciencia Et Technica, No 24, Marzo, 2004.

$$\sum_{\{(i,j)\in A: i\in U, j\in(V-U)\}} x_{ij} \geq 1 \; 2 \leq |U| \leq |V| - 2 \qquad \textbf{(24)}$$

El Problema del Agente Viajero ó TSP (por sus siglas en inglés) ha sido sin duda uno de los problemas más estudiados por su alto campo de acción: administración de depósitos[35], optimización de vías férreas[36], optimización en redes de computación[37], benchmarking[38], simulación de campos eléctricos[39], fabricación de vidrio[40], entre otros. Debido al gran impacto que tiene el TSP en diferentes campos de la ciencia, es objeto de estudio en la presente tesis.

---

[35] Oberlin, P.; Rathinam, S.; Darbha, S. "A transformation for a Multiple Depot, Multiple Traveling Salesman Problem", American Control Conference (ACC '09) , Junio 10-12, 2009.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5160665&isnumber=5159764

[36] Pop, P.C.; Pintea, C.-M.; Pop Sitar, C.; Dumitrescu, D., "A Bio-Inspired Approach for a Dynamic Railway Problem", Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), International Symposium, Septiembre 26-29, 2007.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4438136&isnumber=4438061

[37] Jiu-Sheng Chen; Xiao-Yu Zhang; Jing-Jie Chen, "An elastic net method for solving the traveling salesman problem", Wavelet Analysis and Pattern Recognition (ICWAPR '07), International Conference Noviembre 2-4, 2007.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4420741&isnumber=4420716

[38] Lishan Kang; Aimin Zhou; McKay, B.; Yan Li; Zhuo Kang. "Benchmarking algorithms for dynamic travelling salesman problems," Evolutionary Computation (CEC2004), Congress Junio19-23, 2004.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1331045&isnumber=29392

[39] Chun-Bo Feng; Min Jiang; Jinsong Feng. "Solving traveling salesman problem by simulated electric field method," Intelligent Processing Systems (ICIPS '97), 1997 IEEE International Conference, Octubre 28-31, 1997.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=669214&isnumber=14755

[40] Richard, P.; Falcon Chang, M.; Monmarche, N.; Proust, C., "Visiting the traveling salesman problem with Petri nets and application in the glass industry", Emerging Technologies and Factory Automation (EFTA '96), Proceedings., Noviembre 18-21, 1996
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=573298&isnumber=12362

Los problemas combinatorios pueden ser clasificados en dos grandes grupos considerando la existencia ó no de algoritmos polinomiales para resolver cada tipo de problema. El primero es el problema tipo polinomial (P) para el cual existen algoritmos con esfuerzos computacionales de tipo polinomial para encontrar la solución óptima; y el segundo es el tipo No-Polinomial (NP) para el cual no se conocen algoritmos con esfuerzos computacionales de tipo polinomial para encontrar la solución óptima[41], en la figura 9 puede apreciarse esta definición.

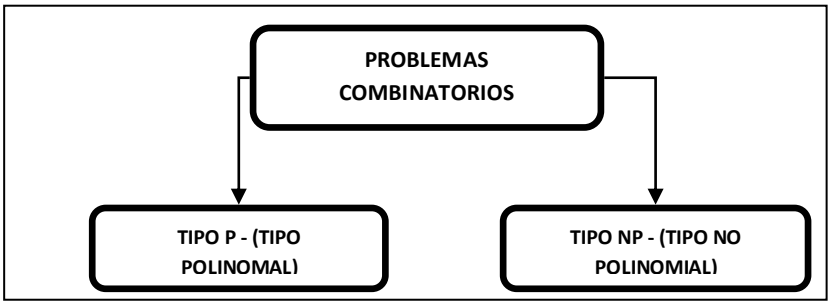

Figura 9. Clasificación de los problemas combinatorios. Los problemas combinatorios pueden ser de tipo P ó NP dependiendo de la existencia de un algoritmo que en tiempo computacional brinde la solución optima.

2.2.1. TECNICAS HEURISTICAS

Las técnicas heurísticas son algoritmos que encuentran soluciones de buena calidad para problemas combinatorios complejos; es decir, para problemas tipo NP. Los algoritmos heurísticos son más fáciles de implementar y encuentran buenas soluciones con esfuerzos computacionales relativamente pequeños; sin embargo, renuncian (desde el punto de vista teórico) a encontrar la solución óptima global de un problema. En problemas de gran

[41] Ibid 34.

tamaño rara vez un algoritmo heurístico encuentra la solución óptima global. Sin embargo, las técnicas heurísticas han sido utilizadas para la solución de problemas combinatorios, por ejemplo: planeación de actividades[42], servicios de ruteo[43], planeación de sistemas[44], optimización en la utilización de espacios[45], entre otros.

2.2.2. TECNICAS METAHEURISTICAS

El termino metaheurística fue acuñado en el mismo artículo que la metaheurtística Búsqueda Tabú (Glover, 1986)[46]. Una metaheurística es una estrategia maestra que permite aproximar las soluciones a las soluciones óptimas de un problema combinatorio.

---

[42] Wen-Xiang Gu; Jin-Li Li; Ming-Hao Yin; Jun-Shu Wang; Jin-Yan Wang, "A novel causal graph based heuristic for solving planning problem," Machine Learning and Cybernetics, 2008 International Conference, Julio 12-15, 2008.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4620775&isnumber=4620699

[43] Xin Yuan; Xingming Liu, "Heuristic algorithms for multi-constrained quality of service routing," INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE , 2001.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=916275&isnumber=19794

[44] Yang Li; Wei-Xiang Gu; Ming-Hao Yin; Yuan Wasng, "Planning system based on heuristic," Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on , Vol 3, pag. 1385-1390, Agosto 18-21, 2005.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1527160&isnumber=32625

[45] Ha, C.; Kuo, W., "Multi-path heuristic for redundancy allocation: the tree heuristic," Reliability, IEEE Transactions on , Vol 55, No 1, pag. 37-43, Marzo, 2006.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1603891&isnumber=33703

[46]Glover, Fred; Laguna, Manuel. "Tabu Search", Kluwer Academic Publishers, pag. 17, Boston – London, 2007.

2.2.2.1. ALGORITMOS GENETICOS[47]

Los Algoritmos Genéticos (AG) son herramientas que imitan a la naturaleza con el fin de resolver problemas complejos partiendo del concepto de la evolución. El primer AG fue desarrollado en los años 60s por John Holland junto con un grupo de colegas y alumnos de la universidad de Michigan. En un principio lo desarrolló para resolver problemas de aprendizaje de máquina. Holland analizó el fenómeno de adaptación en sistemas naturales y artificiales, con lo que pudo obtener un modelo tentativo para usarlo en un sistema computacional[48].

Los AG permiten representar dos interpretaciones:

- **Al nivel del genotipo:** Se refiere a la carga genética hereda por sus antepasados.

- **Al nivel del fenotipo:** Se refiere a las características visibles del individuo.

Los AG ejecutan una búsqueda simultánea en diferentes regiones del espacio factible, realiza una intensificación sobre algunas de ellas y luego explora otros subespacios a través del intercambio de información entre configuraciones. Los mecanismos que utiliza son:

- **Selección:** Los individuos de la población inicial se generan de forma aleatoria y los operadores genéticos trabajan a escala genotípica sobre la representación elegida (binaria, real, entre otros), obteniéndose una nueva generación de la población. El operador genético que permite elegir las configuraciones de la población actual que

[47] Ibid 14

[48] Castro, Salvador. "Creación de Portafolios de Inversión Utilizando Algoritmos Evolutivos Multiobjetivo", Grupo de Computación Evolutiva Departamento de Ingeniería Eléctrica, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México, 2005.

participará en la generación de las configuraciones de la nueva población y termina después de decidir el número de descendientes de cada configuración de la población actual[49].

- **Recombinación:** También llamado crossover, la recombinación es el mecanismo que permite pasar la información genéticas de los individuos, es decir, la información de un par de cromosomas originales a sus descendientes, por lo que puede saltar de un espacio de búsqueda a otro con lo que genera diversidad genética[50].

- **Mutación:** La mutación es el operador secundario y permite realizar la intensificación en un espacio en particular "caminando" a través de vecinos. Lo que hace es intercambiar el valor de un gen de un cromosoma en una población. De manera aleatoria se elige un cromosoma como posible candidato, luego se genera un número aleatorio y si este es menor que la tasa (se elige en el rango [0.001, 0.05]) de mutación ($\rho<\rho m$), entonces es realizada la mutación[51].

El algoritmo que resume los pasos a seguir por un AG se muestran en la figura 10.

Generar la población inicial
Calcular la función de aptitud para cada individuo de la población
Repetir
    Aplicar un operador de selección a los individuos de la población
    Aplicar el operador genético de recombinación a los individuos de la población
    Aplicar el operador genético de mutación a los individuos de la población
Hasta que se cumpla una condición de terminación.

Figura 10. Algoritmo Genético[52]

[49]Ibid 48.
[50]Ibid 48.
[51]Ibid 48.
[52]Ibid 48.

2.2.2.2. RECOCIDO SIMULADO[53]

El Recocido Simulado o Simulated Annealing (SA) es una de las metaheurísticas más clásicas. Su nombre está inspirado en el proceso de recocido del acero y cerámicas que consiste en calentar y luego enfriar lentamente el material para así variar las propiedades físicas del material. Con el calor se aumenta la energía en los átomos para poder desplazar las posiciones iniciales (que sería un mínimo local de energía), y el enfriamiento lento les da mayores probabilidades de recristalizar en configuraciones con menor energía que la energía inicial (que sería el mínimo global). Dada su simplicidad y buenos resultados para diversos tipos de problemas, es una herramienta ampliamente utilizada en diversas áreas. Está basado en el trabajo de 1953 de Metrópolis en el campo de la termodinámica estadística[54].

El Algoritmo Metrópolis en un principio fue modelado por Metrópolis en 1953 como método de estudio de la termodinámica en estadística. Metrópolis modeló el proceso de recocido simulando los cambios energéticos en un sistema de partículas conforme decrece la temperatura, hasta que converge a un estado estable[55]. Según las leyes de la termodinámica a una temperatura *t* la probabilidad de un incremento energético de magnitud *dE* se puede aproximar a:

$$P[dE] = \exp(-dE/kt) \quad \textbf{(25)}$$

---

[53]Ibid 20.

[54]Díaz, Adenso. "Recocido Simulado", Universidad de Oviedo, 2004.

[55]Ibid 54.

Donde *k* es la constante física de Boltzmann.

Surgió como consecuencia de la comparación de los problemas formulados en el campo de la termodinámica con los campo de la investigación de operaciones. El Algoritmo Metrópolis está basado en el "Método Monte-Carlo", el cual estudia las propiedades de equilibrio en el análisis del comportamiento microscópico de los cuerpos. Metrópolis genera una secuencia de estados de un sólido, es decir, dado un sólido en un estado *i* con energía E_i, se genera el siguiente estado *j* mediante la aplicación de un mecanismo que lo transforma al siguiente estado a través de un pequeño disturbio. Ahora la energía será la energía del siguiente estado, o sea, E_j. Entonces si la diferencia de energía $(E_j - E_i)$ es menor o igual a cero, el estado j es aceptado. Si ocurre lo contrario, el estado j se acepta con la probabilidad de acuerdo a la ecuación 26.

$$e\left\{\frac{E_i - E_j}{T \times k}\right\} \qquad \textbf{(26)}$$

Donde T es la temperatura y k es una constante física conocida como constante de Boltzman.

La ecuación 26 se llama Criterio de Metrópolis. Si la disminución de la temperatura es cambiada de manera paulatina, el sólido alcanzará un estado de equilibrio en cada nivel de temperatura. En el algoritmo es necesario generar un gran número de transiciones en un nivel dado de temperatura, para poder encontrar el equilibrio.

A principios de los años 80s, diversas publicaciones ayudaron en el desarrollo de nuevas herramientas para la solución de problemas combinatorios de gran complejidad, tales como

los realizados por Kirkpatrick en 1983[56] y, tal vez uno de los más importante para el área productiva, el hecho por Cerny en 1985[57] para el TSP, con lo que pudieron mostrar cómo el planteamiento de Metrópolis podría ser aplicado a problemas de optimización, asociando conceptos clave del proceso original de simulación, con elementos de optimización combinatoria. Por lo tanto, cualquier implementación de búsqueda local se puede convertir en una implementación de un recocido simulado al elegir elementos del entorno de manera aleatoria y aceptar los movimientos a una solución no mejor de acuerdo a una probabilidad dada por la ecuación 26. La figura 11 muestra el algoritmo básico del recocido simulado para minimización.

```
Sea f(s) el coste de la solución s y sea N(s) su entorno.
Seleccionar una solución inicial s0;
Seleccionar una temperatura inicial t0 > 0;
Seleccionar una función de reducción de la temperatura α;
Seleccionar un número de iteraciones nrep;
Seleccionar un criterio_de_parada;
REPETIR
REPETIR
Seleccionar aleatoriamente una solución s Î N(s0);
Sea δ = f(s) - f(s0);
SI δ < 0 ENTONCES s0 = s
SINO
Generar aleatoriamente u Î U(0,1);
SI u < exp(-δ/t) ENTONCES s0 = s;
FINSINO
HASTAQUE cuenta_iteraciones = nrep
t = α (t);
HASTAQUE criterio_de_parada = CIERTO.
```

Figura 11. Algoritmo Básico del Recocido Simulado para Minimización [58]

El SA resultado del algoritmo Metrópolis es un algoritmo metaheurístico de búsqueda para problemas de optimización global que trata de encontrar una buena aproximación al

[56] Kirkpatrick, S.; Gellatt, C; Vecchi, M. "Optimization by Simulated Annealing", Science, 220: pag. 671-680, 1983.

[57] Cerny, V. "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm", Journal of Optimization Theory and Applications, pag. 41—51, 1985.

[58] Ibid 54.

óptimo. El algoritmo aplica una acción combinada del mecanismo de generación de alternativas y del criterio de aceptación. El parámetro de control es T_k (la temperatura) y N_k el número de alternativas generadas en la *k*-ésima iteración del algoritmo. Cuanto *T* es grande se aceptan grandes deterioros y a medida que *T* se acerca a cero no acepta ningún deterioro. Esta característica en especial hace que el algoritmo SA sea diferente a otro algoritmo con respecto a los óptimos locales.

A partir del estado *i* con costo *f(i)* se genera el estado *j* con costo *f(j)*. El criterio de aceptación determina si este nuevo estado es aceptado con la siguiente probabilidad:

$$P_T\{acepta\ j\} = \begin{cases} 1 & si\ f(j) \leq f(i) \\ e\left\{\dfrac{f(i) - f(j)}{T}\right\} & si\ f(j) > f(i) \end{cases} \qquad \textbf{(27)}$$

La estrategia seguida en SA es partir de una temperatura alta, permitiendo aceptar soluciones de pobre calidad. Luego se disminuye la temperatura y a la vez la posibilidad de aceptar las soluciones peores. Se empieza con una temperatura inicial T_0 y una velocidad de enfriamiento. La temperatura $T_{(k+1)}$ se calcula a partir de T_k y la velocidad de enfriamiento después de haber hecho $N(T_k)$ iteraciones en la temperatura T_k[59].

T_k.

[59] Ibid 54.

2.2.2.3. COLONIA DE HORMIGAS (ACO) [60]

El primer algoritmo desarrollado en el área de Colonia de Hormigas o Ant Colony Optimization (ACO), fue el Ant System (AS), en donde se estudian sistemas artificiales que simulan colonias de hormigas reales. El ACO se utiliza para resolver problemas de optimización combinatoria, planteados como problemas donde el objetivo es encontrar una secuencia óptima de sus elementos. El ACO ha sido de gran utilidad en la planeación de horarios y ruteo en el TSP. EL algoritmo AS desarrollado por Dorigo[61] también se utiliza para resolver el TSP. La figura 12 muestra el pseudocódigo del algoritmo original del AS.

```
Inicio
For t = 1 to Max_Iter do // Max_Iter es el número de iteraciones
   For k = 1 to m do //m es número de hormigas (agentes)
      Repetir hasta que la hormiga k complete su recorrido
          Seleccionar la siguiente ciudad que se va a visitar
      Calcular la longitud del recorrido de la hormiga k
   Actualizar los niveles de feromona
Fin
```

Figura 12. Pseudocódigo del Algoritmo Original del As[62]

El ACO imita el comportamiento normal de las hormigas, en donde las actividades de búsqueda son distribuidas entre agentes con capacidades simples que se representan como hormigas que se comportan como las hormigas reales.

[60] Ibid 21

[61] Ibid 5.

[62] Mendoza, Benito. "Uso del Sistema de la Colonia de Hormigas para Optimizar Circuitos Lógicos Combinatorios", Universidad Veracruzana, México, 2001.

Las hormigas son insectos prácticamente ciegos, de ahí el interés de saber cómo es posible que puedan encontrar rutas más cortas entre el hormiguero y a comida y viceversa y además adaptarse a cambios u obstáculo que se presenten en el ambiente como se muestra en la figura 10. Luego de un estudio exhaustivo fue posible establecer que las hormigas rastrean feromonas dejadas por ellas mismas mientras caminan y es así como logran encontrar los caminos más cortos. Las feromonas son el medio que utilizan para intercambiar información entre los agentes sobre las rutas. Así pues, si una hormiga se desplaza del punto A a un punto B deja un rastro de feromonas en el suelo y es como marcan el camino que tomaron. Las hormigas prefieren tomar caminos con alta feromona, con lo que se explica el porqué son capaces de adaptarse a cambios en el entorno. Cuando una hormiga encuentra un obstáculo en el camino de feromonas, ya no lo podrá seguir por lo que deben elegir un nuevo camino, ir a la derecha o a la izquierda del obstáculo. La elección del camino es de manera aleatoria, pero la probabilidad es que la mitad de las hormigas tomará la derecha y la otra mitad la izquierda. En este punto las hormigas eligieron de forma aleatoria el camino más corto para llegar a la comida y después de un tiempo crearán un depósito de feromona más fuerte que el dejado por las hormigas que eligieron el camino más largo, por lo que pasarán más hormigas por el camino corto, ya que saben que llegan al otro lado más rápido que las demás. Como a medida que pasan más hormigas que encontraron el camino corto, más grande será el depósito de feromonas, pasado el tiempo todas las hormigas que vienen atrás preferirán caminar por esta ruta. En el caso que una hormiga vague sola en forma aleatoria, si encuentra un rastro de feromonas, esta puede decidir con alta probabilidad seguirlo y aumentar el nivel de feromonas y hacerlo más fuerte.

En el TSP una hormiga artificial representa al agente que viaja de ciudad en ciudad. El agente escoge la ciudad a la que irá después de acuerdo a una función de probabilidad que dependerá de la cantidad de “feromonas” dejada en ese camino y de una función heurística definida en función de la distancia.

Se ubican *m* hormigas artificiales en ciudades de forma aleatoria, en cada unidad de tiempo ellas se mueven de una ciudad a otra actualizando el rastro de feromonas en los caminos. Esto es llamado una actualización local de rastro. Cuando todas las hormigas completan el recorrido, la hormiga que realizó el camino más corto realiza una nueva actualización de feromonas, pero solo en los caminos que usó, lo que es llamado actualización global de rastro y depende del inverso de la distancia recorrida por la hormiga[63].

2.2.2.4. BÚSQUEDA TABÚ (TS)[64]

Es un procedimiento metaheurístico utilizado para manejar un algoritmo heurístico de búsqueda tabú (TS) local y así evitar que el proceso se detenga en un óptimo local. Por lo tanto, TS realiza una exploración a través del espacio de configuraciones delimitando adecuadamente los óptimos locales.

Para evitar que el proceso regrese a los óptimos locales y entre en un ciclo repetitivo, la búsqueda tabú clasifica los movimientos más recientes como "movimientos tabú"; estos prohíben que una configuración sea visitada nuevamente. Este método tiene dos tipos de memoria: Memorias de corto plazo y largo plazo. La de corto plazo contiene los eventos ocurridos recientemente y en la de largo plazo, se almacenan los datos de frecuencia de determinados eventos. La información de la memoria de largo plazo es fundamental para definir las estrategias de diversificación, las cuales permiten explorar otras regiones no visitadas anteriormente.

[63]Ibid 63.

[64]Ibid 27.

Cuando un movimiento ha sido clasificado como tabú y después de ser analizado produce una función objetivo mejor que un valor de referencia escogido (que puede ser una incúmbete u otra buena solución previamente encontrada), entonces se aplica la denominada regla de aspiración, esta consiste en cancelar la prohibición y aceptar el movimiento.

2.2.2.5. GRASP[65]

Es una evolución de los algoritmos heurísticos constructivos, especialmente de aquellos que usan indicadores de sensibilidad. Con estos indicadores se calcula la variación de la función objetivo con respecto a las variables de interés del problema de optimización, y se usan para identificar los atributos atractivos de tal problema. Emplea una propuesta intermedia entre Simulated Annealing y Búsqueda Tabú para realizar la fase de exploración y ha mostrado ser eficiente en la solución de problemas complejos de optimización.

Para un problema genérico, el algoritmo GRASP tiene los siguientes pasos:

1. Implementar una fase de pre-procesamiento.

2. Realizar la fase de búsqueda tabú constructiva.

[65] Tupia, Manuel. "Un Algoritmo GRASP para Resolver el Problema de la Programación de Tareas Dependientes en Máquinas Diferentes (Task Scheduling)", Universidad Nacional Mayor de San Marcos, Lima, Perú, 2005.

3. Realizar la fase exploratoria y actualizar la mejor solución encontrada si se supera la solución a comparar.

4. Si el criterio de parada no se satisface volver al paso 2. Si no, finalizar el proceso. La respuesta del algoritmo es la mejor solución almacenada.

Con el pre-procesamiento se trata de identificar los atributos más interesantes del problema con los que se realiza el proceso de búsqueda tabú. Esto permite disminuir el espacio de soluciones que se quiere explorar. La fase de búsqueda tabú constructiva consiste en encontrar una solución de calidad para el problema con base en un algoritmo heurístico constructivo donde se escoge, en cada paso, un elemento de una lista de tamaño k denominada RCL, donde se clasifican las variables más atractivas y se obtiene una incúmbete de buena calidad del problema.

La fase constructiva del algoritmo presenta los siguientes pasos:

1. Escoger una solución inicial que puede ser vacía, es decir, sin adicionar variables la cual se transforma en la solución actual del problema.

2. Para la solución actual del problema, elaborar una lista que clasifica las mejores k variables que identifican el indicador de sensibilidad.

3. Escoger en forma aleatoria o probabilística una de las variables de la lista y actualizar la solución con la adición o sustracción de la variable escogida.

4. Si la solución actual es factible o se satisface el criterio de parada, se finaliza la fase constructiva.

La fase exploratoria procura encontrar una solución óptima local en la vecindad de la solución de la fase constructiva.

2.3. AUTOMATAS

Intuitivamente, un autómata es un dispositivo teórico capaz de procesar una secuencia finita de símbolos de un alfabeto, cambiando su estado si procede. De entre los posibles estados que puede alcanzar un autómata se distinguen los aceptadores que indican el reconocimiento, por parte del autómata, de la secuencia tratada.[66]

2.3.1. AUTOMATA FINITO DETERMINISTA

Formalmente un autómata finito determinista es una 5 – tupla:

$$D = (Q, \Sigma, \delta, q_0, F) \quad \textbf{(28)}$$

[66] Castro, David. "Teoría de autómatas y lenguajes formales y gramática", Universidad de Alcalá, 2004.

En donde:

Q, es un conjunto no vacío, finito de estados.

Σ, es un alfabeto finito.

$\delta: Q \times \Sigma \rightarrow Q$, es la función de transición.

$q_0 \in Q$, es el estado inicial.

$F \subseteq Q$, es el conjunto de estados de finalización o aceptadores.

La definición anterior muestra la vista estática del autómata más no su comportamiento, la definición del función de transición nos dirá cual será el comportamiento que asumirá el autómata de acuerdo al alfabeto de entrada, para esto deberemos extender la función δ al dominio $Q \times \Sigma^*$.

2.3.1.1. FUNCION DE TRANSICION EXTENDIDA

Dado un autómata finito determinista $D = (Q, \Sigma, \delta, q_0, F)$, se define la extensión al dominio $Q \times \Sigma^*$, (denotada por $\hat{\delta}$) de manera inductiva en la longitud de cadenas:

Dado que $q \in Q$, definimos $\hat{\delta}(q, \varepsilon) = q$

Dados $q \in Q$, $w \in \Sigma^*$ y $\sigma \in \Sigma$, definimos $\hat{\delta}(q, w\sigma) = \hat{\delta}\big(\hat{\delta}(q, w), \sigma\big)$

Una característica esencial de los autómatas finitos deterministas, es que la función de transición extendida está definida para todos los elementos de $Q \times \Sigma$, por lo tanto, esto obliga a todos los elementos que pertenecen al conjunto de estados finitos ($q \in Q$) a estar interconectados por medio de alguna letra del alfabeto de entrada con otro estado de este mismo conjunto. Esto indica que para cualquier letra del alfabeto de entrada:

$$\delta(q, \sigma) = p \mid q, p \in Q \ \wedge \ \sigma \in \Sigma \qquad \textbf{(29)}$$

2.3.1.2. REPRESENTACION DE AUTÓMATAS FINITOS DETERMINISTAS MEDIANTE DIAGRAMAS DE TRANSICIÓN[67]

Una representación de los Autómatas Deterministas es la dada por los diagramas de transición. En ella se representa el autómata mediante un grafo orientado sujeto a las siguientes consideraciones:

1. Cada nodo del grafo tiene asociado de manera univoca un estado del autómata.

2. Dos nodos, digamos p y q están unidos mediante un arco dirigido de p a q si y solo si, existe un símbolo del alfabeto digamos $\sigma \in \Sigma$ tal que $\delta(p, \sigma) = q$ dicho arco se etiqueta σ.

[67]Ibid 27.

3. Los nodos correspondientes a estados aceptadores (finalizadores) se representan mediante una doble circunferencia con el fin de distinguirlos de los restantes.

4. Al nodo correspondiente al estado inicial llega una flecha sin origen concreto.

Ejemplo 1. Considerar, por ejemplo, el autómata $A = (Q, \Sigma, \delta, q_0, F)$, donde $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0,1\}$, $F = \{q_2\}$ y la función de tranisción δ, se define mediante la tabla 7.

Tabla 7. Función de transiciónδ. Función de transición δ para el autómata del ejemplo 1.

	0	1
$\rightarrow q_0$	q_2	q_0
q_1	q_1	q_1
$*\, q_2$	q_2	q_1

Nótese en la tabla 7 que el estado inicial suele indicarse mediante una flecha mientras que, los estados finales, se indican con un asterisco.

El diagrama de acuerdo a las especificaciones dadas puede apreciarse en la figura 13.

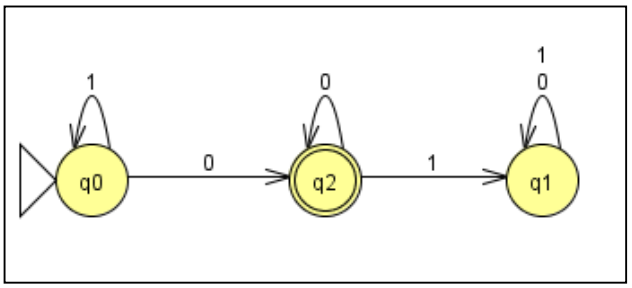

Figura 13. Representación de un autómata finito determinista mediante el uso de diagramas de transición. Representación mediante diagramas de transición del AFD del ejemplo 1.

2.3.2. AUTOMATA FINITO DETERMINISTA DE INTERCAMBIO[68]

Un Autómata Finito Determinista de Intercambio (AFDI), es un grafo que permite representar el espacio de soluciones factibles de un problema de naturaleza combinatoria en los cuales el orden de los elementos importa y no se permiten repeticiones. Formalmente, un AFDI se define como:

$$M = (\overline{X_0}, f(\bar{X}), Q, \Sigma, \delta, q_0, F) \quad \textbf{(30)}$$

Donde:

$\overline{X_0}$ es el vector que contiene el orden original de los elementos asociados al problema representado.

[68] Ibid 4

f($\overline{X}$) es la función que se desea optimizar en el problema modelado.

Q, es el conjunto de estados que conforman el aútomata, cada estado $q_k \in Q$ representa una solución al problema modelado. La estructura de cada estado viene conformada de la siguiente manera:

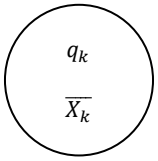

Figura 14. Estructura de un estado de un AFD – I. Cada estado que compone un AFDI está compuesto por un orden específico de los elementos modelados.

$\overline{X_k}$, corresponde a un orden específico de los elementos asociados al problema modelado y así mismo se convierte en una solución factible al problema propuesto.

Σ es el alfabeto finito de entrada. Este contiene todas las posibles combinaciones entre los índices del vector de entrada agrupándolos en parejas, debido a que estos son distintos, las parejas de Σ vendrían dadas por tuplas que cumplen la condición:

$$\{(k,j)|k = 1,2 \dots n, j = k+1, k+2, .., n\} \quad \textbf{(31)}$$

Donde n es el número de elementos del vector de entrada.

δ, es la función de transición la cual toma un estado q, $q \in Q$, y una tupla de Σ y devuelve un nuevo estado t, $t \in Q$, esto es $\delta(q,(i,k)) = t$, en donde $(i,k) \in \Sigma$. Es importante resaltar que, debido a que los AFD – I provienen de los AFD , por 3.2.1.1, cada estado $q \in Q$ debe tener transición con todos los símbolos de Σ. q_0 es el estado inicial del AFD – I. Este contiene una solución inicial para el problema modelado. F, es el conjunto de estados de finalización, para los AFD – I todos los elementos de Q son elementos de F.

Ejemplo 2. Supóngase el vector de entrada $\overline{X} = (1,2,3)$ y la función objetivo:[69]

$$f(\bar{X}) = 0{,}3 * x_1 + 0{,}2 * x_2 + 0{,}1 * X_3 \qquad \textbf{(32)}$$

Para la construcción del AFDI equivalente a estos datos, procedemos a hacer lo siguiente:

1. **Establecer el estado inicial:** En este caso nuestro estado inicial q_0 contendrá el vector de entrada $\bar{X} = (1,2,3)$, por lo tanto $\bar{X} = \overline{X_0}$.

2. **Establecer el conjunto Σ:** Por la ecuación 31 sabemos que $\Sigma = \{(1,2),(1,3),(2,3)\}$.

3. **Establecer la función δ:** Debido a que los AFDI exigen que los estados tengan transiciones con todos los símbolos del alfabeto, comenzamos por el estado inicial el cual es q_0, cada estado nuevo lo nombramos como q_k, luego hacemos transiciones con cada q_k hasta agotarlos todos como podemos apreciarlo en la tabla 8.

[69] Ibid 4.

Tabla 8. Función δ para el AFD I con vector de entrada $\overline{\mathrm{X}} = (1,2,3)$

$\delta(q_0,(1,2)) = (2,1,3) = \boldsymbol{X_3} = \boldsymbol{q_3}$	$\delta(q_0,(1,3)) = (3,2,1) = \boldsymbol{X_5} = \boldsymbol{q_5}$	$\delta(q_0,(2,3)) = (1,3,2) = \boldsymbol{X_1} = \boldsymbol{q_1}$
$\delta(q_1,(1,2)) = (2,3,1) = \boldsymbol{X_4} = \boldsymbol{q_4}$	$\delta(q_1,(1,3)) = (3,1,2) = \boldsymbol{X_2} = \boldsymbol{q_2}$	$\delta(q_1,(2,3)) = (1,2,3) = \boldsymbol{X_0} = \boldsymbol{q_0}$
$\delta(q_2,(1,2)) = (3,2,1) = \boldsymbol{X_5} = \boldsymbol{q_5}$	$\delta(q_2,(1,3)) = (1,3,2) = \boldsymbol{X_1} = \boldsymbol{q_1}$	$\delta(q_2,(2,3)) = (2,1,3) = \boldsymbol{X_3} = \boldsymbol{q_3}$
$\delta(q_3,(1,2)) = (1,2,3) = \boldsymbol{X_0} = \boldsymbol{q_0}$	$\delta(q_3,(1,3)) = (2,3,1) = \boldsymbol{X_4} = \boldsymbol{q_4}$	$\delta(q_3,(2,3)) = (3,1,2) = \boldsymbol{X_2} = \boldsymbol{q_2}$
$\delta(q_4,(1,2)) = (1,3,2) = \boldsymbol{X_1} = \boldsymbol{q_1}$	$\delta(q_4,(1,3)) = (2,1,3) = \boldsymbol{X_3} = \boldsymbol{q_3}$	$\delta(q_4,(2,3)) = (3,2,1) = \boldsymbol{X_5} = \boldsymbol{q_5}$
$\delta(q_5,(1,2)) = (3,1,2) = \boldsymbol{X_3} = \boldsymbol{q_3}$	$\delta(q_5,(1,3)) = (1,2,3) = \boldsymbol{X_0} = \boldsymbol{q_0}$	$\delta(q_5,(2,3)) = (2,3,1) = \boldsymbol{X_4} = \boldsymbol{q_4}$

Como puede observarse en la tabla 8 todos los estados tienen transiciones con todos los símbolos del alfabeto y en consecuencia el AFD – I contiene todas las posibles permutaciones para los elementos del vector de entrada $\overline{\mathrm{X}}$.

4. **Establecer los valores obtenidos para cada evaluación de los $\overline{\boldsymbol{X_k}}$ en la función objetivo.** El siguiente paso es calcular el valor de la evaluación en la función objetivo de cada $\overline{\mathrm{X_k}}$, con lo cual obtenemos la tabla 9.

Tabla 9. Evaluación de los vectores en la función objetivo

Estado $\boldsymbol{q_k}$	Vector $\overline{X_k}$	Valor $\boldsymbol{f}(\overline{\boldsymbol{X_k}})$
$\boldsymbol{q_0}$	(1,2,3)	1
$\boldsymbol{q_1}$	(1,3,2)	1.1
$\boldsymbol{q_2}$	(3,2,1)	1.4
$\boldsymbol{q_3}$	(2,1,3)	1.1
$\boldsymbol{q_4}$	(2,3,1)	1.3
$\boldsymbol{q_5}$	(3,2,1)	1.4

El AFDI de la figura 15 representa el autómata de este ejemplo. Nótese que cada estado tiene transición con todos los elementos de Σ lo cual representa el intercambio llevado a cabo en el vector $\overline{X_k}$ asociado al estado q_k para alcanzar el nuevo estado q_t cuyo vector es $\overline{X_t}$.

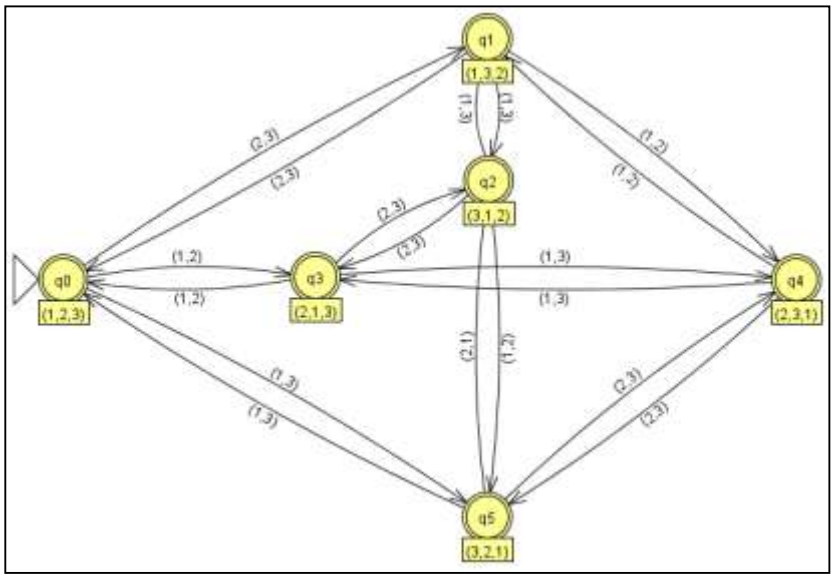

Figura 15. Representación del AFDI para el vector de entrada $\overline{\mathbf{X}} = (\mathbf{1}, \mathbf{2}, \mathbf{3})$. Cada estado representa una solución factible.

2.3.3. AUTOMATA FINITO DETERMINISTA MULTIOBJETIVO

Un Autómata Finito Determinista Multi-objetivo (AFDM) es una estructura de datos que permite el modelado del espacio de soluciones factibles de un problema combinatorio bajo un esquema multi-objetivo.

Formalmente, un AFDM se define como:

$$M = (\ Q, \Sigma, \delta, Q_0, F(X)) \qquad \textbf{(33)}$$

2.3.2.1. Q Conjunto de Estados

Q es el conjunto de Estados del AFDM. Cada estado Q contiene un vector solución X factible del problema combinatorio modelado. X contiene un orden de las variables decisión del problema combinatorio.

La estructura de un estado $q \in Q$ se aprecia en la figura 16.

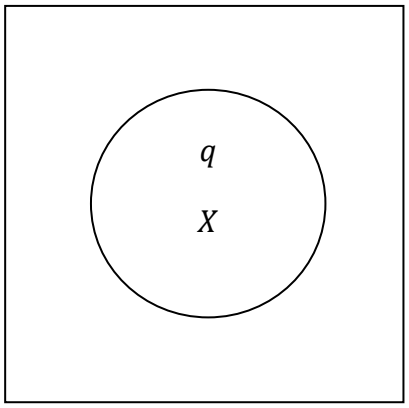

Figura 16. Estructura de un estado $q \in Q$**.** Cada estado posee una solución X distinta.

No existen dos estados con el mismo X.

Definición 1. $\boldsymbol{X_q}$. Se define X_q como el vector solución X del estado $q \in Q$.

Ejemplo 1: Sea una empresa que tiene tres máquinas que realizan un trabajo cualquiera, se tienen tres procesos $P_1, P_2\ y\ P_3$ cuya duración es de 10, 50 y 5 minutos respectivamente, si los procesos se ejecutan en paralelo y pueden utilizar cualquiera de las máquinas disponibles, ¿De cuantas maneras se pueden asignar las máquinas a procesos distintos para que se ejecuten?

Fácilmente esta situación puede presentarse a diario en cualquier empresa de producción que trabaje con máquinas y procesos en paralelo, supongamos que el vector $\overline{\boldsymbol{P}} = (\boldsymbol{P_k}, \boldsymbol{P_i}, \boldsymbol{P_j})$ representa la solución máquina 1 ejecuta $\boldsymbol{P_k}$, máquina 2 ejecuta $\boldsymbol{P_i}$, y máquina 3 ejecuta $\boldsymbol{P_j}$, por lo tanto el conjunto de estados Q que representa el conjunto de soluciones factibles asociados a este problema son los representados en la figura 17.

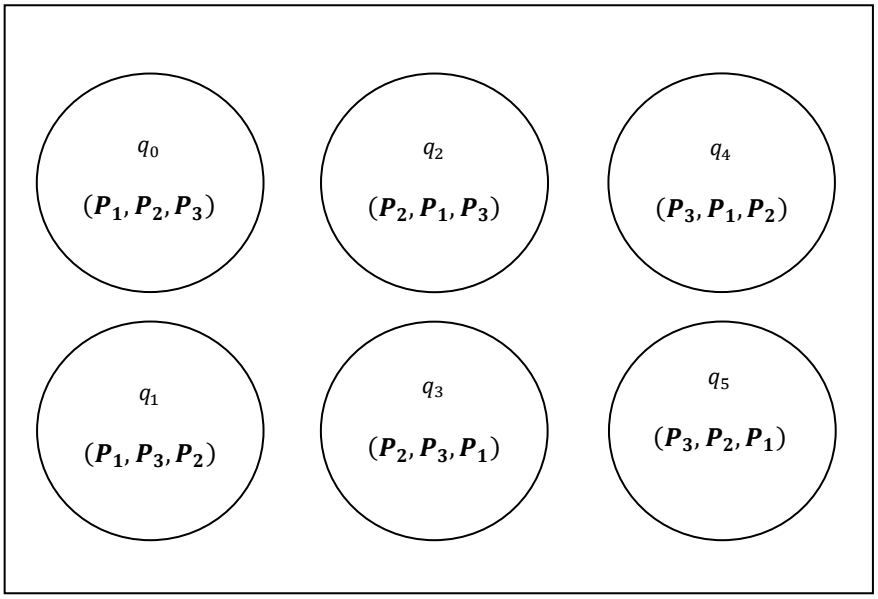

Figura 17. Representación del espacio de soluciones factibles para el problema del ejemplo 1. Cada estado posee una manera diferente de asignar las máquinas a los procesos.

Como puede apreciarse en la figura 17, por la definición 1 tenemos que $X_{q0} = (P_1, P_2, P_3), X_{q1} = (P_1, P_3, P_2),\ X_{q2} = (P_2, P_1, P_3),\ X_{q3} = (P_2, P_3, P_1),\ X_{q4} = (P_3, P_1, P_2)$ y $X_{q5} = (P_3, P_2, P_1)$.

Debido a que en cada estado X_q es diferente, y como en un AFDM el número de estados es igual al número de todas las posibles soluciones, si no existen restricciones en el problema, existen $n!$ estados, esto es:

$$\#_{estados}(Q) = n! \qquad \textbf{(34)}$$

2.3.3.2. Σ Alfabeto Finito de Entrada

El conjunto Σ se conoce como el alfabeto finito de entrada. Σ está compuesto por los elementos que cumplen la condición:

$$\{(k, j) | k = 1,2 \ldots n, j = k + 1, k + 2, .., n\} \qquad \textbf{(35)}$$

Esto implica que Σ contiene todas las posibles maneras de intercambiar los elementos de un vector. Consecuentemente, Σ indica todas las posibles maneras de intercambiar los elementos de cualquier X_q.

Teorema 1. Dado un AFDM cuyos vectores solución son de norma n, el número de elementos del conjunto Σ es igual a:

$$\#_{elementos}(\Sigma) = \frac{n \times (n-1)}{2} \quad \textbf{(36)}$$

Demostración 1.

Por la definición de subgrupos, si agrupamos los elementos en grupos de a dos (k = 2) , tendríamos la siguiente expresión:

$$nC_2 = \frac{n!}{(n-2)! \times 2!} \quad \textbf{(37)}$$

Aplicando la definición de factorial de manera recurrente en la ecuación 37 se obtiene:

$$nC_2 = \frac{(n-2)! \times (n-1) \times n}{(n-2)! \times 2!} \quad \textbf{(38)}$$

Cancelando los $(n-2)!$ y reemplazando el valor de 2! = 2 en la ecuación 38 se obtiene:

$$nC_2 = \frac{(n-1) \times n}{2} \quad \textbf{(39)}$$

Y con ello queda demostrado el teorema 1.

Ejemplo 2. El conjunto Σ para el problema del ejemplo 1 es:

$$\Sigma = \{(P_1, P_2), (P_1, P_3), (P_2, P_3)\} \quad \textbf{(40)}$$

La cantidad de elementos concuerda con la ecuación la propuesta por el teorema 1:

$$\#_{elementos}(\Sigma) = \frac{3 \times (3-1)}{2} = 3 \quad \textbf{(41)}$$

2.3.3.3. δ Función de transición

La función de transición δ describe el comportamiento del AFDM. δ permite el desplazamiento entre los diversos estados del AFDM utilizando elementos de Σ, esto es:

$$\boldsymbol{\delta(q, a) = r/q, r \in Q \wedge q \neq r \wedge a \in \Sigma} \quad \textbf{(42)}$$

Como puede apreciarse, la función descrita en 42, obtiene un elemento de Σ para pertubar los elementos del vector X_q, es decir, intercambiar los elementos del vector y consecuentemente, caer en otro estado con solución X_r. La función es definida formalmente como:

$$\boldsymbol{\delta}: \boldsymbol{Q} \rightarrow \boldsymbol{Q} \qquad \textbf{(43)}$$

Ejemplo 3. Retomando el ejemplo 1, la función δ definida para el estado q_0 es: Sobre cada nuevo estado destino obtenido al aplicar la función δ (en este caso q_1,q_2 y q_5) , debe aplicarse la misma función con el objetivo de alcanzar nuevos estados. El proceso termina cuando no se obtienen nuevos estados. Una vez se han obtenido todos los estados, se puede afirmar que se ha recorrido el espacio de soluciones factibles del problema modelado por medio del AFDM, lo que equivaldría a una búsqueda exhaustiva.

Tabla 10. Función δ para q_0 del AFDM del ejemplo 1. Debido a que los AFDM provienen de los AFD, todos los estados deben tener transición con todos los elementos del alfabeto.

Estado origen q	**Elemento de Σ**	**$\delta(q,a)/\ q \in Q a \in \Sigma$**	**Estado destino r**
q_0	(P_1, P_2)	$\boldsymbol{\delta}\big(q_0, (P_1, P_2)\big)$	q_2
q_0	(P_1, P_3)	$\boldsymbol{\delta}\big(q_0, (P_1, P_3)\big)$	q_5
q_0	(P_2, P_3)	$\boldsymbol{\delta}\big(q_0, (P_2, P_3)\big)$	q_1

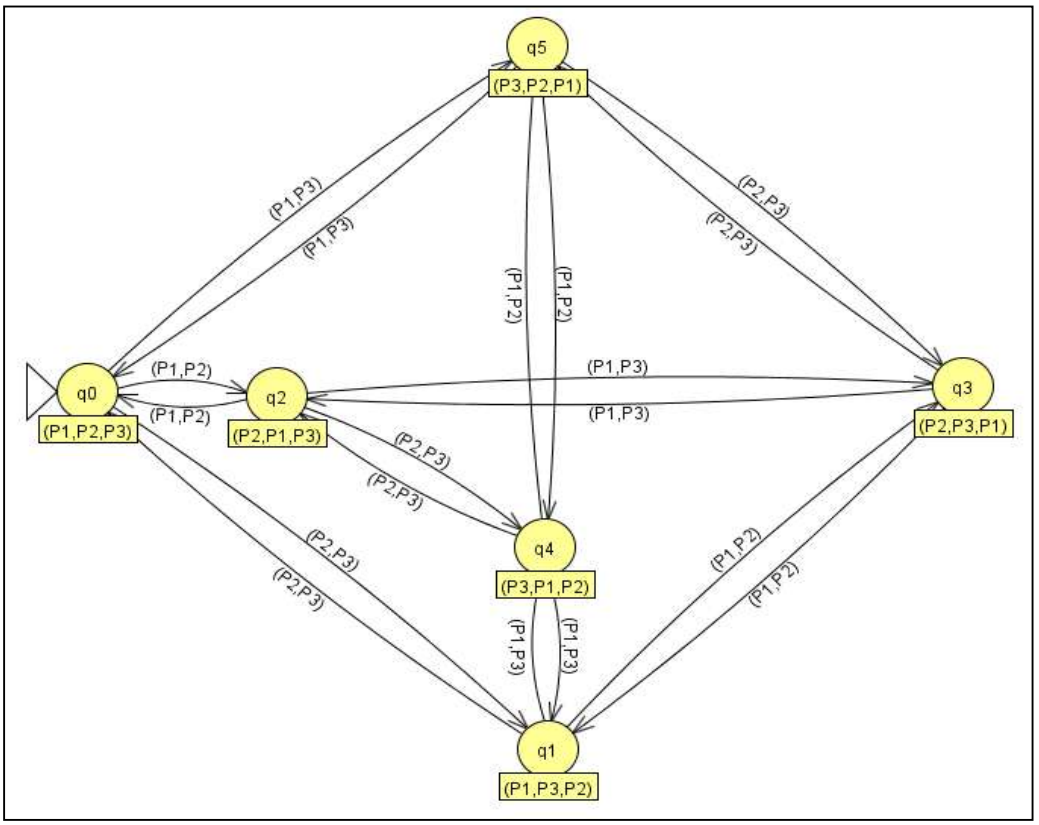

Figura 18. AFDM construido a partir del estado inicial q_0 del ejemplo 1. El AFDM es construido al aplicar la definición de δ sobre cada nuevo estado encontrado.

El autómata de la figura 18 ilustra el autómata construido a partir de la función δ. La flecha sobre q_0 indica que el autómata comenzó a generarse desde ese estado.

2.3.3.4. Q_0 Conjunto de Estados Iniciales.

El conjunto de estados iniciales Q_0 contiene los estados por el cual se comenzará a generar el autómata. En otras palabras, Q_0 representa las soluciones iniciales del problema. Así

pues en el ejemplo 3 Q_0 solo contenía un elemento igual al estado q_0.

2.3.3.5. $F(X)$ Conjunto de Funciones Objetivos.

La función $F(X)$ es el conjunto de funciones objetivos del problema, si el problema tiene n objetivos, $F(X)$ es:

$$\boldsymbol{F(X) = \{f_1(X), f_2(X), \dots, f_n(X)\}} \quad \textbf{(44)}$$

Esta función se encuentra definida formalmente como:

$$\boldsymbol{F: R^n \rightarrow R^m \mid n \geq 1 \wedge m \geq 2} \quad \textbf{(45)}$$

$F(X)$ recibe como argumento de entrada una solución X_q. En 45, n representa el número de variables decisión del problema combinatorio y m representa el número de funciones objetivo. Por lo tanto $F(X)$ toma las variable decisión y las evalúa en cada función objetivo devolviendo una m – tupla.

Ejemplo 4. Supóngase el siguiente conjunto de funciones bi – objetivo (2 objetivos):

$$F(X) = \left\{ f_1(X) = \sum_{i=1}^{3} i * X_i \, , f_2(X) = \sum_{i=1}^{3} \frac{1}{i} * X_i \right\} \quad \textbf{(46)}$$

Como puede apreciarse en 46, este es un problema bi – objetivo. Si se evalúa el espacio de soluciones factibles del problema 1 en 46, se obtienen los resultados de la tabla 11. El FP para este problema puede apreciarse en la figura 19.

Tabla 11. Función $F(X)$ aplicada a los estados q_i, $q_i \in Q$, del AFDM del ejemplo 1. En este caso en particular $F(X)$ recibe el orden en que se asignarán los trabajos en las tres máquinas y devuelve dos valores correspondientes a cada función objetivo.

Estado	Asignación			Tiempos			$F(X)$	
q_i	M1	M2	M3	M1	M2	M3	$f_1(X)$	$f_2(X)$
q_0	P1	P2	P3	10	50	5	125	36.66
q_1	P1	P3	P2	10	5	50	170	29.16
q_2	P2	P1	P3	50	10	5	85	56.66
q_3	P	P3	P1	50	5	10	90	55.83
q_4	P3	P1	P2	5	10	50	175	26.66
q_5	P3	P2	P1	5	50	10	135	33.33

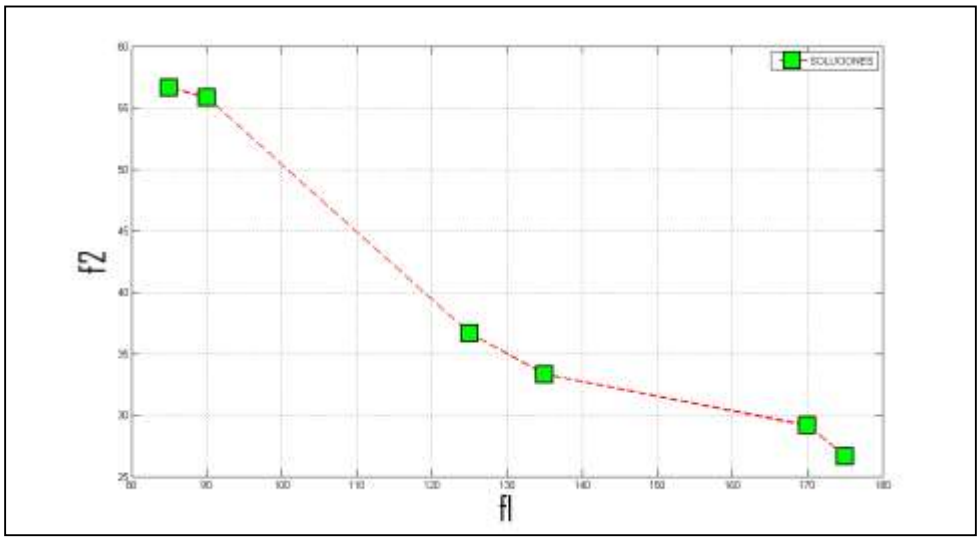

Figura 19. Conjunto de soluciones factibles para el problema del ejemplo 1. Conjunto de soluciones factibles para el problema del ejemplo 1 al utilizar la función 48, en verde se encuentran las soluciones.

3. METAHEURISTICA PROPUESTA

3.1. DISEÑO E IMPLEMENTACION DE UNA METAHEURISTICA HIBRIDA BASADA EN RECOCIDO SIMULADO, ALGORITMOS GENETICOS Y TEORIA DE AUTOMATAS PARA LA OPTIMIZACION BI-OBJETIVO DE PROBLEMAS COMBINATORIOS

Los Autómatas Finitos Deterministas Multi-objetivo (AFDM) permiten modelar y describir de manera efectiva el espacio de soluciones factibles parar problemas combinatorios. Sin embargo, esto no garantiza encontrar soluciones óptimas debido a que la cota de crecimiento del número de estados de Q es exponencial respecto al número de variables de decisión ó entrada. Es decir que el espacio de soluciones factibles que resulta de un AFDM es muy grande por lo que es imposible encontrar soluciones óptimas dentro del conjunto. Es por ello que es necesario formular nuevas metaheurística que permitan reducir el espacio de soluciones factibles a un espacio de soluciones óptimas o *"vectores no dominados"*. La creación de nuevas técnicas constituye un aporte invaluable a la industria donde será posible reducir el uso de insumos no renovables, además del hecho de incitar a otros a la investigación de este campo.

Al aumentar el número de objetivos a optimizar dentro de un proceso productivo, de igual forma aumentará la complejidad de encontrar el conjunto de soluciones óptimas, por lo que se busca el diseño una herramienta que permita solucionar el problema planteado por medio del uso de un autómata finito determinista multi-objetivo que optimice problemas del tipo tri-objetivo utilizando las técnicas de recocido simulado y algoritmos genéticos con lo que se busca diversificar la población y evitar caer en óptimos locales.

El marco básico de trabajo (framework) de la metaheurística propuesta llamada AERSMIDA se explica a continuación:

1. $\boldsymbol{crearSolucionesIniciales}(\boldsymbol{A})$. Este método especifica los estados de Q que conformarán Q_0. Lógicamente Q es un conjunto implícito dado su gran tamaño (espacio de soluciones factibles). Por lo tanto, crear los estados que contienen las soluciones iníciales, equivaldrá a especificar los estados que conformarán Q que conformarán Q_0. Las soluciones iníciales pueden ser creadas de acuerdo al problema que se está solucionando. Las soluciones obtenidas por técnicas heurísticas pueden ser una buena aproximación a las soluciones óptimas. Partir de un buen conjunto de soluciones iníciales puede ayudar a la técnica a obtener óptimos con menor esfuerzo computacional.

2. $\boldsymbol{mejorarSolucionesIniciales}(\boldsymbol{Q_0})$**.** Este método opcional permite mejorar las soluciones que fueron encontradas inicialmente. Esto se puede llevar a cabo ya sea aplicando métricas a las soluciones obtenidas inicialmente ó replanteando el modelo original mediante problemas mono – objetivos para optimizar cada uno de los objetivos. Con esto se expande aún más el conocimiento de Q y con ello los posibles estados del AFDM donde las soluciones son óptimas.

3. $\boldsymbol{explorarAFDM}(\boldsymbol{Q_\varphi}, \boldsymbol{Q_*}, \boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\rho}, \boldsymbol{A})$**.** Esta última etapa de la metaheurística determina el modo de explorar el AFDM. De manera formal, este método puede ser apreciado en la figura 20. A continuación, se explican cada uno de los parámetros (variables) de entrada:

Entradas:

$\boldsymbol{\theta}$ Máximo número de iteraciones.

$\boldsymbol{\beta}$ Máxima cantidad de transiciones por estado.

$\boldsymbol{\alpha}$ Máxima cantidad de transiciones por estado sin mejora.

$\boldsymbol{\rho}$ Máxima cantidad de estados analizados por ciclo.

$\boldsymbol{A}$ Autómata Finito Determinista Multi – objetivo.

Salida:

$\boldsymbol{Q_{\varphi}}$ Conjunto de estados con soluciones no dominadas.

AERSMIDA$(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\rho}, \boldsymbol{A})$

1 $\boldsymbol{Q_{\varphi}} \leftarrow \boldsymbol{\phi}$ */*Se inicializa el conjunto de soluciones en vacio*/*

2 $\boldsymbol{Q_0} \leftarrow \boldsymbol{crearSolucionesIniciales}(\boldsymbol{A})$ */*Soluciones iníciales*/*

3 $\boldsymbol{Q_0} \leftarrow \boldsymbol{mejorarSolucionesIniciales}(\boldsymbol{Q_0})$ */*Opcional*/*

4 $\boldsymbol{Q_{\varphi}} \leftarrow \boldsymbol{Q_0}, \boldsymbol{Q_*} \leftarrow \boldsymbol{\phi}$ */*Inicialización de Conjuntos*/*

5 $\boldsymbol{explorarAFDM}(\boldsymbol{Q_{\varphi}}, \boldsymbol{Q_*}, \boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\rho}, \boldsymbol{A})$ */*Núcleo del Algoritmo*/*

6 **AERSMIDA**$\leftarrow \boldsymbol{Q_{\varphi}}$

END AERSMIDA

Figura 20. Framework de la Metaheurística AERSMIDA

Donde $\boldsymbol{Q_{\varphi}}$ es un conjunto que contiene cada uno de los estados que conforman el conjunto solución del algoritmo. Al final, este conjunto contiene los estados cuyas soluciones conforman la aproximación al Frente de Pareto Óptimo. La condición con la cual se acepta un estado en este conjunto es que su solución no se encuentre dominada por alguna solución correspondiente a un estado perteneciente al conjunto. $\boldsymbol{Q_*}$ es el conjunto contiene los estados cuyas soluciones determinan la dirección del FPO. La condición necesaria para que un estado pertenezca a este conjunto es, además de su solución no ser dominada por ningún elemento de $\boldsymbol{Q_{\varphi}}$, dominar por lo menos a una solución de un estado en $\boldsymbol{Q_{\varphi}}$, $\boldsymbol{\theta}$ es un valor numérico entero que determina el máximo número de iteraciones de la metaheurística. $\boldsymbol{\rho}$ describe la cantidad de estados seleccionados aleatoriamente por iteración. $\boldsymbol{\beta}$ es un valor numérico entero que indica el máximo número de iteraciones por cada estado seleccionado aleatoriamente. $\boldsymbol{\alpha}$ Indica el máximo número de iteraciones

permitidas por estado sin presentar mejoras. $\boldsymbol{A}$ Es el AFDM que modela el problema combinatorio. Este especifica, los conjuntos Q, Q_0 y Σ, la función de transición δ y las funciones objetivos $F(X)$.

AERSMIDA evita caer en óptimos locales al utilizar dos técnicas que lo evitan y a la vez permiten una diversificación de la población: aplicando el operador genético de recombinación de la población, es decir a realizar el cruzamiento y la técnica de recocido simulado. Los pasos que sigue para realizar el procedimiento son los siguientes:

- Selecciona un número X de soluciones factibles del conjunto de soluciones factibles Q_s (las cuales están almacenadas en forma de vector), y los almacena en Q_c.

- Selecciona de manera aleatoria dos soluciones del conjunto Q_c y los cruza, de acuerdo con las teorías de algoritmos genéticos, aplicando el operador genético de recombinación de la población. Luego de tomar las dos soluciones, realiza una partición aleatoria en la k-ésima posición, para tomar la cabeza de la primera solución (padre) y combinarla con la cola de la segunda solución (madre), como se muestra en la figura 21 según la teoría de algoritmo genético. Como no se admiten repeticiones, en caso de que al realizar el cruzamiento suceda, el algoritmo automáticamente llenará los espacios del vector con otras ciudades que aún no hayan sido visitadas de manera aleatoria. Las soluciones obtenidas se almacenan en un nuevo conjunto llamado Q_m. El cruzamiento evita caer en óptimo locales al diversificar la población de soluciones.

- Toma una solución del conjunto Q_m y lo perturba de acuerdo con la teoría de Recocido Simulado, basándose en el algoritmo Simulated Annealing Metaheuristic

of Deterministic Swapping (SAMODS), desarrollado por Sarabia[70], como se muestra en la figura 22. Las soluciones las almacena en el conjunto Q_{φ}.

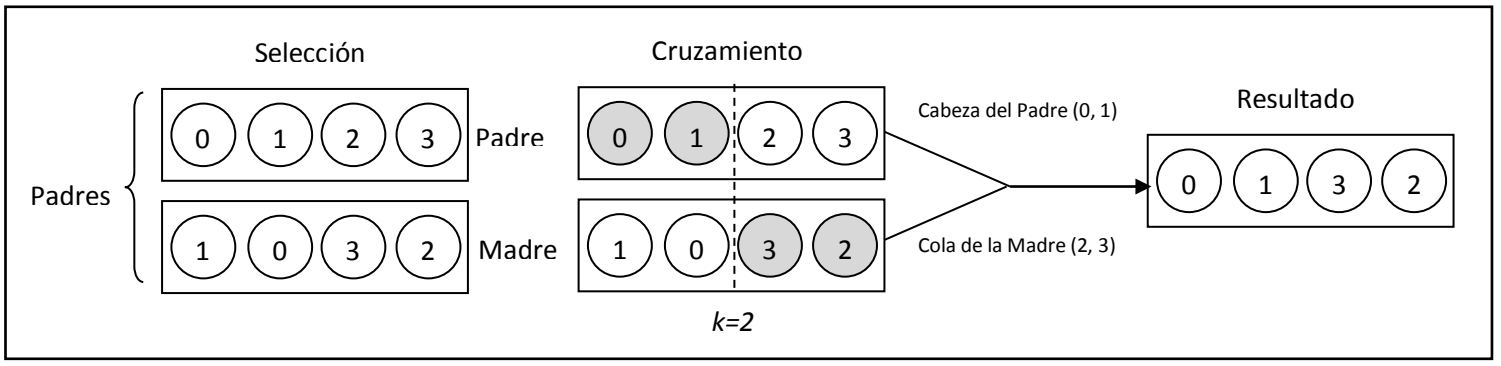

Figura 21. Cruzamiento Realizado en AERSMIDA

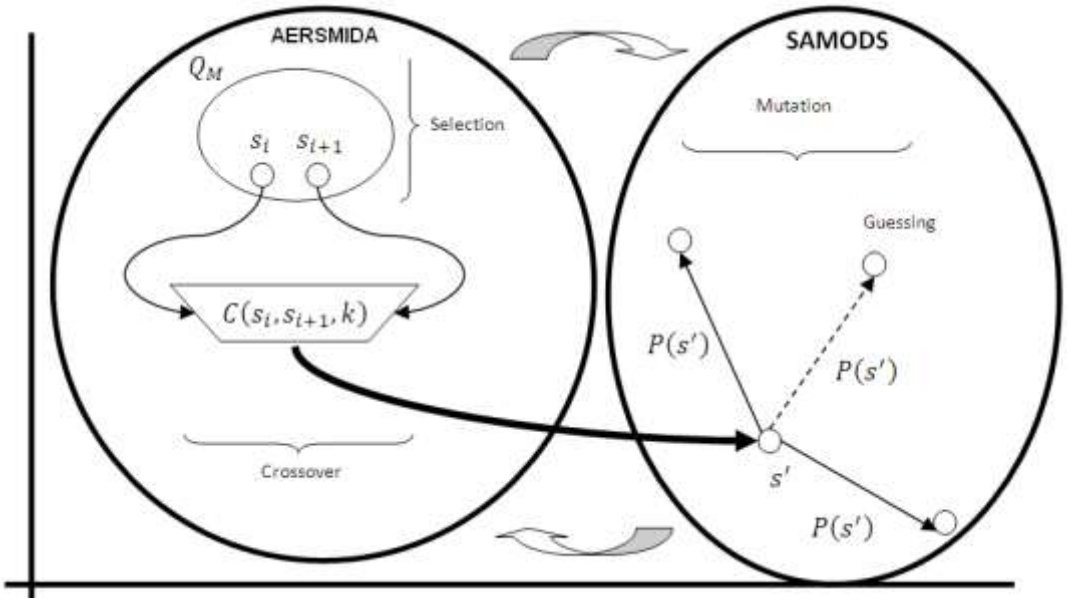

Figura 22. AERSMIDA utilizando SAMODS como estrategia de búsqueda local

[70]Sarabia, Justo. "Diseño e Implementación de una Metaheurística Basada en Autómatas Finitos Deterministas y Técnica de Recocido Simulado para la Optimización Bi-Objetivo de Problemas Combinatorios". Universidad del Norte, Barranquilla, Colombia, 2010

Cuando se obtiene las soluciones iniciales con o sin mejoras, se recorre el Autómata Finito Determinista Multi-Objetivo (AFDM), con el fin de buscar estados o soluciones no dominadas, de la siguiente manera:

1. Hacer Frente de Pareto Óptimo (FPO) como el conjunto de soluciones iniciales.

2. Para un número máximo de iteraciones establecidos θ haga:

3. Seleccione aleatoriamente un número ρ de estados del conjunto Q_{φ}.

4. Para cada estado seleccionado, aplique β veces la función δ en busca de Soluciones No Dominadas en Q_{φ}. Todos los estados q encontrados cuyos X_q no sean dominados por ningún $X_r / r \in Q_{\varphi}$ se agregan a Q_{φ}. Los elementos q cuyos X_q dominan por lo menos a un elemento $X_r / r \in Q_{\varphi}$ se agregan al Conjunto de Soluciones Elitista Q_*. Se seleccionan elementos en caso de que pasen α iteraciones sin que una solución seleccionada en el paso 3 presente mejoras, teniendo en cuenta que $\#_{estados}(Q_*) > 0$.

5. Remover los elementos dominados de Q_{φ} y Q_*. Ir a 2.

3.2. ANALISIS DE LA COMPLEJIDAD DE AERSMIDA

Asumamos valores muy grandes para θ, β, ρ y α:

$$\alpha = \beta = \rho = \theta = n \qquad \textbf{(47)}$$

Con *n* muy grande al igual que el resto de variables.

Analizaremos la complejidad se hará en el peor de los casos. En la figura del marco básico de trabajo (framework) de la Metaheurística AERSMIDA, las líneas 1,4 y 6 suman 4 por las cuatro asignaciones realizadas. Las líneas 2,3 y 5 son sub-ecuaciones a calcular. Así pues, la ecuación inicial del análisis será:

$$T_{AERSMIDA}(n) = 4 + T_{crearSoluciones}(n) + T_{mejorarSoluciones}(n) + T_{explorarAFDM}(n) \quad \textbf{(48)}$$

Como *n* es un valor muy grande para todos los parámetros utilizados, las funciones $T_{crearSoluciones}(n)$ y $T_{mejorarSoluciones}(n)$**,** asumiendo que crear un estado con su respectiva solución corresponden a una unidad de tiempo, equivalen a:

$$T_{crearSoluciones}(n) + T_{mejorarSoluciones}(n) = \sum_{i=1}^{n} 1 + \sum_{i=1}^{n} 1 \quad \textbf{(49)}$$

Resolviendo las sumatorias de 47 se obtiene:

$$T_{crearSoluciones}(n) + T_{mejorarSoluciones}(n) = 2n \quad \textbf{(50)}$$

En la figura que muestra un framework del método *explorar AFDM*, en el método $T_{explorarAFDM}(n)$, se inicia con un ciclo en la línea 1 pero el ciclo encierra otro. Al final del primer ciclo encontramos dos métodos que permiten remover soluciones dominadas de conjunto de estados – soluciones, como son *n* soluciones remover dominados implica contrastar cada solución contra el resto, asumiendo que comparar la solución de un estado con el de otro consume una unidad de tiempo, la ecuación inicial para $T_{explorarAFDM}(n)$ es:

$$T_{explorarAFDM}(n) = \sum_{k=1}^{n}\left(\sum_{j=1}^{n}? + 2\sum_{u=1}^{n}\sum_{v=1}^{n}1\right) \quad \textbf{(51)}$$

Al resolver doble sumatoria interna en 51 se obtiene:

$$T_{explorarAFDM}(n) = \sum_{k=1}^{n}\left(\sum_{j=1}^{n}? + 2n^2\right) \quad \textbf{(52)}$$

La ecuación 52 se convierte en:

$$T_{explorarAFDM}(n) = \sum_{k=1}^{n}\left(\sum_{j=1}^{n}\left(2 + \sum_{i=1}^{n}?\right) + 2n^2\right) \quad \textbf{(53)}$$

La ecuación 53 se convierte en:

$$T_{explorarAFDM}(n) = \sum_{k=1}^{n}\left(\sum_{j=1}^{n}\left(2 + \sum_{i=1}^{n}\left(6 + \sum_{p=1}^{n}1\right)\right) + 2n^2\right) \quad \textbf{(54)}$$

Resolviendo las sumatorias de 54, se obtiene:

$$T_{explorarAFDM}(n) = \sum_{k=1}^{n}(2n + 8n^2 + n^3) = 2n^2 + 8n^3 + n^4 \quad \textbf{(55)}$$

Reemplazando se obtiene:

$$T_{AERSMIDA}(n) = 4 + 2n + 2n^2 + 8n^3 + n^4 \quad \textbf{(56)}$$

Organizando por coeficientes, la ecuación de tiempo en el peor de los casos para AERSMIDA es:

$$T_p(n) = n^4 + 8n^3 + 2n^2 + 2n + 4 \quad \textbf{(57)}$$

Por lo tanto la cota superior de la metaheurística es igual a:

$$O\left(T_p(n)\right) = max\{n^4, 8n^3, 2n^2, 2n, 4\} = O(n^4) \quad \textbf{(58)}$$

A partir de 58, se puede obtener la cota inferior de AERSMIDA, el análisis es simple:

$$T_p(n) = n^4 + 8n^3 + 2n^2 + 2n + 4 \geq 8n^3 + 2n^2 + 2n + 4 \geq 2n^2 + 2n + 4 \geq 2n + 4 \quad \textbf{(59)}$$

Por lo tanto de 59 se concluye que:

$$\Omega\left(T_p(n)\right) = \Omega(n + 4) \quad \textbf{(60)}$$

4. EXPERIMENTACION Y ANALISIS DE RESULTADOS

Para poder analizar de manera certera la efectividad de la metaheurística AERSMIDA, se realizaron pruebas con instancias del Problema del Agente Viajero Simétrico (TSP), el cual es considerado un problema No Polinomial Complejo o Difícil (NP-hard) y su planteamiento puede ser utilizado en gran variedad de problemas en la industria.

Dado que se necesita comprobar la efectividad del método frente a otros que tratan el mismo problema, se tomaron instancias de la librería TSPLIB[71]. Dicha librería contiene más de 200 de instancias del TSP y unas 500 instancias de problemas conocidos tales como el Ruteo de Vehículos el cual es un derivado del TSP. TSPLIB es administrada por el Grupo de Investigación de Optimización Discreta de la Universidad de Heidelberg en Alemania. Una gran cantidad de trabajos científicos relacionados con este tipo de investigación utilizan estas instancias para probar la efectividad de sus métodos y compararlos con otros autores relacionados con el tema[72].

Tomando en cuenta lo anterior, el TSPLIB constituye una buena opción para probar y contrastar los resultados obtenidos por AERSMIDA con otros métodos.Las instancias utilizadas de esta librería se encuentran en la tabla 12.

Los resultados obtenidos son contrastados contra otras técnicas utilizando las métricas presentadas en el marco teórico (Generación de Vectores no Dominados, Espaciamiento, Distancia Generacional e Inversa de la Distancia Generacional).

[71] TSPLIB. http://comopt.ifi.uni-heidelberg.de/index.html
[72] TSPLIB. http://comopt.ifi.uni-heidelberg.de/publications/index.html

Tabla 12. Instancias de la TSPLIB utilizadas en las pruebas de AERSMIDA.

Nombre de la Instancia	Número de Ciudades	Número de Objetivos
KROAB100	100	2
KROAC100	100	2
KROAD100	100	2
KROAE100	100	2
KROBC100	100	2
KROBD100	100	2
KROBE100	100	2
KROCD100	100	2
KROCE100	100	2
KRODE100	100	2

4.1. ESPECIFICACIÓN DEL AFDM PARA LA PRUEBA

Para llevar a cabo la prueba primero es necesario definir el AFDM que permite modelar el Problema del Agente Viajero.

Siendo Q el espacio de soluciones factibles del TSP, cada estado $q_i \in Q$ contiene un X_{q_i} que representa un camino para recorrer el conjunto de ciudades. Como el conjunto tiene un tamaño grande, este se encuentra de manera implícita en la memoria.

Σ representa las 2 – tuplas que permiten intercambiar los elementos de un vector, para intercambiar las ciudades de una solución y obtener nuevas soluciones, es decir explorar nuevos estados de Q.

δ es una función que toma una 2 – tupla de Σ e intercambia de posición los elementos del vector solución de un estado. Esto es con el objetivo de alterar el orden de visita de las ciudades y así obtener mayor diversidad en las soluciones encontradas.

Q_0 es el conjunto de soluciones iníciales.

$F(X)$ son las *n* matrices donde se encuentra las respectivas distancias entre cada una de las ciudades, siendo *n* el número de objetivos del problema. Las distancias son Euclidianas, se toma el orden de visita de las ciudades de X y se suman las respectivas distancias recorridas de la matriz.

4.2. COMPARACIÓN DE LOS RESULTADOS DE AERSMIDA CON LA METAHEURÍSITICA MIDA

Los resultados obtenidos con AERSMIDA, fueron contrastados con los resultados de otras metaheurísticas que resuelven el problema del agente viajero. La técnica con la que se contrastarán los resultados obtenidos es con la metaheurística MIDA.

El Ingeniero Elías Niño, es el autor e implementador de la Metaheurística de Intercambio Determinista sobre Autómatas (MIDA). Para comparar MIDA corrió 20 veces cada uno de los algoritmos, ya que es el número de veces que se corrieron las pruebas hechas por otros algoritmos con los cuales comparó sus resultados, y recolectó las soluciones no dominadas de todas las corridas. Para que la comparación equitativa, las soluciones no dominadas propuestas por AERSMIDA fueron recolectadas de la misma forma.

Para comparar los resultados de las instancias KROAB100, KROAC100, KROAD100, KROAE100, KROBC100, KROBD100, KROBE100, KROCD100, KROCE100 y KRODE100, por medio de las métricas ESPACIAMIENTO, GD e IGD, se tomaron los resultados obtenidos por Niño en el artículo "*MODS: A Novel Metaheuristic of*

Deterministic Swapping for the Multi-Objective Optimization of Combinatorials Problems"[73].

Los valores de los parámetros de AERSMIDA para ejecutar la prueba son: $\theta = 1000, \rho = 40, \beta = 1000$ y $\alpha = 500$. La figura 23 ilustra los FP de cada algoritmo para cada una de las instancias. La tabla 13 muestra los valores para las métricas GVND, ESPACIAMIENTO, GD e IGD en cada una de las instancias.

Tabla 13. Comparación de los resultados obtenidos por AERSMIDA y MIDA con las instancias KROAB100, KROAC100, KROAD100, KROAE100, KROBC100, KROBD100, KROBE100, KROCD100, KROCE100 y KRODE100, utilizando las métricas GVND, ESPACIAMIENTO, GD e IGD

INSTANCIA	ALGORTIMO	GVND	ESPACIMIENTO	GD	IGD
KROAB100	MIDA	289	0,019288813	14,9446741	2200,00643
	AERSMIDA	8479	0,00069021	0,0193521	3,17541823
KROAC100	MIDA	217	0,03401434	19,1198701	2451,13186
	AERSMIDA	7023	0,000794386	0,02794342	5,48380314
KROAD100	MIDA	281	0,013906191	11,9972745	1807,15845
	AERSMIDA	6289	0,00156113	0,03196484	6,42579018
KROAE100	MIDA	283	0,053267007	13,2513869	2183,78396
	AERSMIDA	6440	0,001295607	0,02791734	5,01919415
KROBC100	MIDA	298	0,01582224	13,7767561	2436,03282
	AERSMIDA	6919	0,001454019	0,03398684	7,99217244
KROBD100	MIDA	241	0,023908639	14,6074	2088,48994
	AERSMIDA	5934	0,001386099	0,03708265	8,15998072
KROBE100	MIDA	280	0,030883013	13,3954499	2424,67171
	AERSMIDA	5802	0,002458221	0,03677847	7,84811075
KROCD100	MIDA	286	0,018445239	11,8873292	1834,38784
	AERSMIDA	6301	0,00140016	0,02880871	5,22946373
KROCE100	MIDA	224	0,02848476	12,6392586	1737,24731
	AERSMIDA	4613	0,002570326	0,00176736	0,01440589
KRODE100	MIDA	228	0,047748166	16,0102025	1720,44948
	AERSMIDA	7745	0,001189943	0,02597846	5,22694781

[73] Niño, Elias ; Ardila, Carlos; Jabba, Daladier; Barrios, Agustín; Donoso, Yesid. "MODS: A Novel Metaheuristic of Deterministic Swapping for the Multi-Objective Optimization of Combinatorials Problems", Computer Technology and Application, Volume 2, Number 4. ISSN 1934-7340, Abril, 2011.

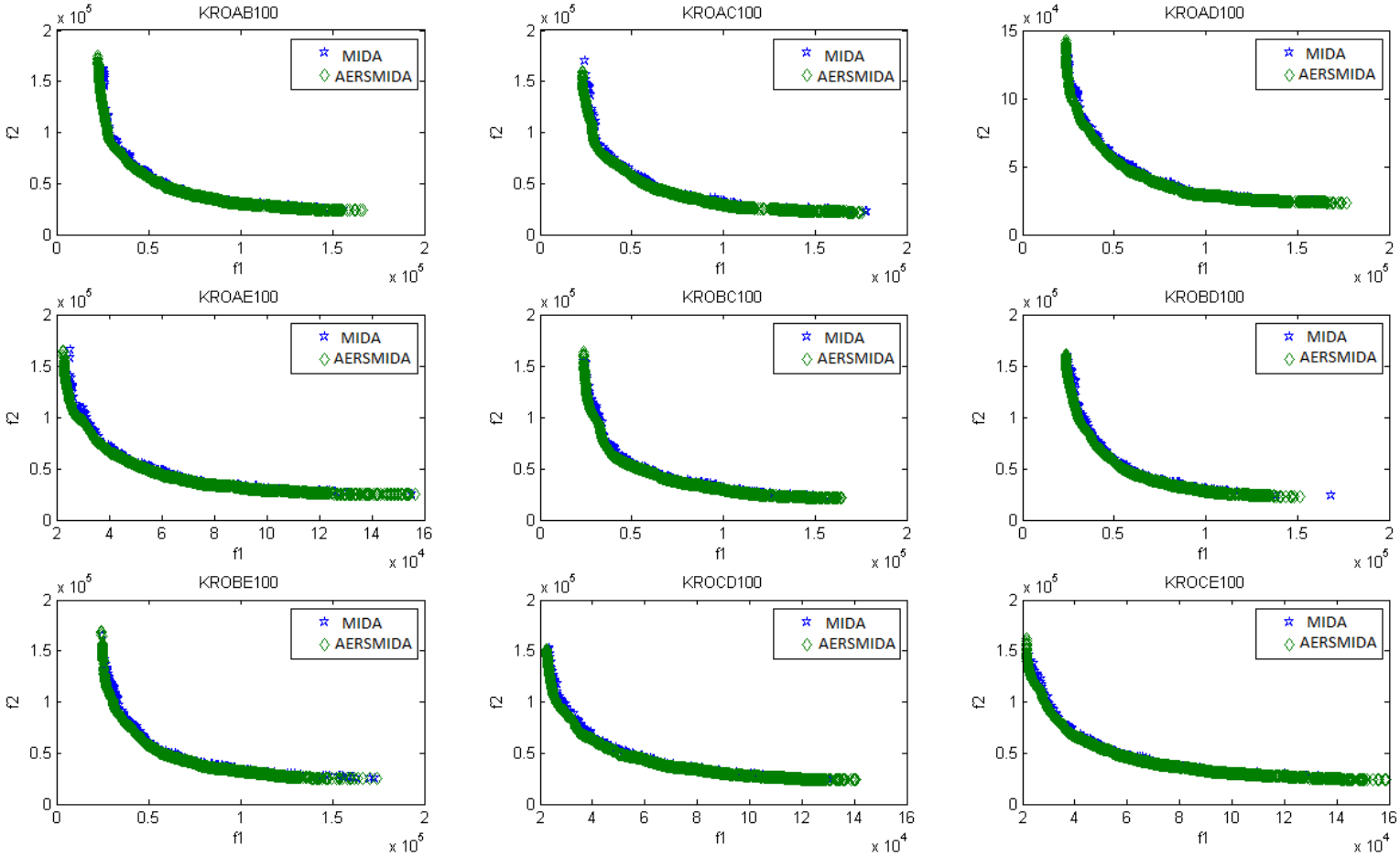

Figura 23. Contraste Visual del FP obtenido por AERSMIDA con las instancias KROAB100, KROAC100, KROAD100, KROAE100, KROBC100, KROBD100, KROBE100, KROCD100, KROCE100 y KRODE100, contra los FPs obtenidos por la metaheurística MIDA

Se busca que GVND (Generación de Vectores no Dominados) sea un número grande, que el Espaciamiento sea un número pequeño al igual que la distancia generacional (GD) y la distancia generacional inversa (IGD).

Como puede apreciarse en la tabla 13, los resultados obtenidos con AERSMIDA tienen mejor desempeño con las métricas GVND, ESPACIAMIENTO, GD e IGD que MIDA, mostrando una superioridad notable en los resultados para las instancias KROAB100, KROAC100, KROAD100, KROAE100, KROBC100, KROBD100, KROBE100, KROCD100, KROCE100 y KRODE100.

5. CONCLUSIONES Y TRABAJOS FUTUROS

Las conclusiones que resultan del trabajo de investigación presentado son citadas a continuación.

1. Se diseñó e implementó una metaheurística basada en Autómatas Finitos Deterministas Multi – Objetivo, algoritmos genéticos y recocido simulado (AERSMIDA) que permite dar solución al problema del agente viajero simétrico bi-objetivo. Al basarse en la teoría de AFDM, las soluciones obtenidas con el método no son del tipo infactible.

2. Al contrastar los resultados con la metaheurística MIDA con dos objetivos usando las instancias KROAB100, KROAC100, KROAD100, KROAE100, KROBC100, KROBD100, KROBE100, KROCD100, KROCE100 y KRODE100, AERSMIDA logra superarla en todas las métricas.

3. La implementación AERSMIDA es un aporte significativo para el sector industrial, dado que permite mejorar los resultados de la única metaheurística basada en autómatas finitos de intercambio determinista implementada y reconocida hasta el momento, por lo constituye un avance significativo en el uso eficiente de recursos.

Los trabajos que se proponen a futuro son:

1. Escribir un artículo ISI con las pruebas realizadas para resolver el Problema del Agente Viajero con las instancias utilizadas en la investigación.

2. Formular una mejora para AERSMIDA que permita resolver otros problemas combinatorios multi – objetivo tales como el ruteo de vehículos, el problema de la mochila y la p – mediana.

3. Contrastar los tiempos obtenidos por AERSMIDA contra los consumidos por otras técnicas.

4. Formular AERSMIDA para una mayor cantidad de funciones objetivos y diferentes instancias.

6. REFERENCIAS BIBLIOGRAFICAS

- Bandyopadhyay, S.; Saha, S.; Maulik, U.; Deb, K., "A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA," Evolutionary Computation, IEEE Transactions, Vol 12, No 3, pag. 269-283, June 2008.
  URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4358775&isnumber=4531586

- Borbulya, István. "An EC-Memory based Method for the Multi-Objective TSP", Proceedings of the 9th annual conference on Genetic and evolutionary computation, New York, 2007.

- Borgulya, István. "An Evolutionary Algorithm for the bi-objective QAP", Computational Intelligence, Theory and Applications Advances, springer series, part 22, pag. 577-586, 2006.

- Cagnina, Leticia. "Optimización Mono y Multiobjetivo a través de una Heurística de Inteligencia Colectiva", Facultad de Ciencias Físico Matemáticas y Naturales, Universidad Nacional de San Luis, Argentina, 2010.

- Cardoso, Pedro; Jesús, Mario; Marquez, Alberto. "MONACO- Multi-Objective Network Optimization based on an ACO", Proc. X Encuentros de Geometría Computacional, Sevilla, España, Junio 16–17, 2003.

- Castro, David. “Teoría de autómatas y lenguajes formales y gramática”, Universidad de Alcalá, 2004.

- Castro, Salvador. “Creación de Portafolios de Inversión Utilizando Algoritmos Evolutivos Multiobjetivo”, Grupo de Computación Evolutiva Departamento de Ingeniería Eléctrica, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México, 2005.

- Cerny, V. “Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm”, Journal of Optimization Theory and Applications, pag. 41—51, 1985.

- Chun-Bo, Feng; Min, Jiang; Jinsong, Feng. "Solving traveling salesman problem by simulated electric field method," Intelligent Processing Systems (ICIPS '97), 1997 IEEE International Conference, Octubre 28-31, 1997. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=669214&isnumber=14755

- Czyzak, P.; Jaszkiewicz, A. “Pareto simulated annealing - a metaheuristic technique for multiobjective combinatorial optimization,” Journal of Multi-Criteria Decision Analysis,Vol 7, No 1, pag. 34–37, Diciembre, 1998.

- Díaz, Adenso. “Recocido Simulado”, Universidad de Oviedo, 2004.

- Doerner, Karl; Gutjahr, Walter; Hartl, Richard; Strauss, Christine; Stummer, Christian. “Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection”, Annals of Operations, pag. 79–99, 2004.

- Doerner, Karl; Hartl, Richard; Reimann, Marc. “Are COMPETants more competent for problem solving?—The Case of Full Truckload Transportation”, Central European Journal of Operations Research, pag. 115–141, 2003.

- Donoso, Yezid; Fabregat, Ramón. “Multi-Objective Optimization in Computer Networks Using Metaheuristics”, Aurbach Publications, pag. 1 – 3, New York, Estados Unidos, 2007.

- Dorigo, Marco; Maniezzo, Vitorio; Colorni, Alberto. “The Ant System: Optimization by a Colony of Cooperating Agents”, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions, Vol 26, No 1, pag. 29-41, Febrero, 1996.

- Gambardella, Luca; Taillard, Éric; Agazzi, Giovanni. “MACS-VRPTW: A Multiple Colony System For Vehicle Routing Problems With Time Windows”, New Ideas in Optimization, McGraw-Hill, pag.. 73–76, 1999.

- Gen, Mitsuo; Cheng, Runwei. “Genetic Algorithms & Engineering Optimization”, Wiley-Interscience Publications, pag. 97-106, New York, Estados Unidos, 2000.

- Glover, Fred; Laguna, Manuel. “Tabu Search”, Kluwer Academic Publishers, pag. 17, Boston – London, 2007.

- Groselj, B.; Malluhi, Q.M., "Combinatorial optimization of distributed queries," Knowledge and Data Engineering, IEEE Transactions, Vol 7, No 6, pag. 915-927, Diciembre, 1995.
  URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=476497&isnumber=10156

- Ha, C.; Kuo, W., "Multi-path heuristic for redundancy allocation: the tree heuristic," Reliability, IEEE Transactions on , Vol 55, No 1, pag. 37-43, Marzo, 2006. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1603891&isnumber=33703

- Hincapie, Ricardo; Rios, Carlos; Gallego, Ramón. "Técnicas Heurísticas aplicadas al problema del cartero viajante (TSP)", Sciencia Et Technica, No 24, Marzo, 2004.

- Iredi, Steffen ; Merkle, Daniel; Middendorf, Martin. "Bi-criterion optimization with multi colony ant algorithms", First International Conference on Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science, pag 359–372, 2001.

- Jiu-Sheng Chen; Xiao-Yu Zhang; Jing-Jie Chen, "An elastic net method for solving the traveling salesman problem", Wavelet Analysis and Pattern Recognition (ICWAPR '07), International Conference Noviembre 2-4, 2007. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4420741&isnumber=4420716

- Kirkpatrick, S.; Gellatt, C; Vecchi, M. "Optimization by Simulated Annealing", Science, 220: pag. 671-680, 1983.

- Lim, A.; Fan Wang. "Multi-depot vehicle routing problem: a one-stage approach", Automation Science and Engineering, IEEE Transactions, Vol 2, No 4, pag. 397-402, Octubre, 2005. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1514459&isnumber=32436

- Lishan Kang; Aimin Zhou; McKay, B.; Yan Li; Zhuo Kang. "Benchmarking algorithms for dynamic travelling salesman problems," Evolutionary Computation (CEC2004), Congress Junio19-23, 2004. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1331045&isnumber=29392

- Mariano, Carlos; Morales, Eduardo. “A multiple objective Ant-Q algorithm for the design of water distribution irrigation networks”, Technical Report HC-9904, Instituto Mexicano de Tecnología del Agua, Mexico, Junio, 1999.

- Mendoza, Benito. “Uso del Sistema de la Colonia de Hormigas para Optimizar Circuitos Lógicos Combinatorios”, Universidad Veracruzana, México, 2001.

- Niño, Elías; Ardila, Carlos. “Algoritmo Basado en Autómatas Finitos Deterministas para la obtención de óptimos globales en problemas de naturaleza combinatoria”, Revista de Ingeniería y Desarrollo, No 25, pag. 100 – 114. ISSN 0122 – 3461, 2009.

- Niño, Elias ; Ardila, Carlos; Jabba, Daladier; Barrios, Agustín; Donoso, Yesid. “MODS: A Novel Metaheuristic of Deterministic Swapping for the Multi-Objective Optimization of Combinatorials Problems", Computer Technology and Application, Volume 2, Number 4. ISSN 1934-7340, Abril, 2011.

- Niño, Elías; Ardila, Carlos; Jabba, Daladier; Donoso, Yesid. “A novel Algorithm based on Deterministic Finite Automaton for solving the mono-objective Symmetric Traveling Salesman Problem”. International Journal of Artificial Intelligence, North America, 2010.

- Oberlin, P.; Rathinam, S.; Darbha, S. "A transformation for a Multiple Depot, Multiple Traveling Salesman Problem", American Control Conference (ACC '09) , Junio 10-12, 2009.
  URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5160665&isnumber=5159764

- Pantoja, Freddy. “Optimización del Proceso de la Amalgamación en la Pequeña Minería del Oro: Mejora de su Recuperación y Disminución de las Perdidas de Mercurio", Universidad Autónoma de Madrid, Madrid, España.

- Pop, P.C.; Pintea, C.-M.; Pop Sitar, C.; Dumitrescu, D., "A Bio-Inspired Approach for a Dynamic Railway Problem", Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), International Symposium, Septiembre 26-29, 2007. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4438136&isnumber=4438061

- Richard, P.; Falcon Chang, M.; Monmarche, N.; Proust, C., "Visiting the traveling salesman problem with Petri nets and application in the glass industry", Emerging Technologies and Factory Automation (EFTA '96), Proceedings., Noviembre 18-21, 1996.
  URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=573298&isnumber=12362

- Sarabia, Justo. “Diseño e Implementación de una Metaheurística Basada en Autómatas Finitos Deterministas y Técnica de Recocido Simulado para la Optimización Bi-Objetivo de Problemas Combinatorios”. Universidad del Norte, Barranquilla, Colombia, 2010

- Torralba, Jose. "Optimización de un proceso de metalurgia de polvos. Análisis experimental de los factores influyentes y sus interacciones", Universidad Politécnica de Madrid, Madrid, España.

- Tovar, Luis; Coronell, Margarita; Donoso Yezid. “Optimización multiobjetivo en redes ópticas con transmisión Multicast, utilizando algoritmos evolutivos y lógica difusa”. Ingeniería & Desarrollo. Número 21. Enero-Junio 2007.

- Tse Guan Tan; Hui Keng Lau; Teo, J. "Cooperative coevolution for pareto multiobjective optimization: An empirical study using SPEA2", TENCON 2007 - 2007 IEEE Region 10 Conference , Octubre 30 a Noviembre 2, 2007. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4429134&isnumber=4428770

- TSPLIB. http://comopt.ifi.uni-heidelberg.de/index.html

- TSPLIB. http://comopt.ifi.uni-heidelberg.de/publications/index.html

- Tupia, Manuel. "Un Algoritmo GRASP para Resolver el Problema de la Programación de Tareas Dependientes en Máquinas Diferentes (Task Scheduling)", Universidad Nacional Mayor de San Marcos, Lima, Perú, 2005.

- Ulungu, E.; Teghem, J.; Fortemps, P.; Tuyttens, D. "MOSA method: A tool for solving multiobjective combinatorial optimization problems," Journal of Multi-Criteria Decision Analysis, vol. 8, no. 4, pp. 221–236, 1999.

- Wang, F.K.; Richards, G.W. "Using combinatorial designs to construct partial concentrators", Communications, IEEE Transactions, Vol 39, No 7, pag 1141-1146, Julio, 1991.

- Wen-Xiang Gu; Jin-Li Li; Ming-Hao Yin; Jun-Shu Wang; Jin-Yan Wang, "A novel causal graph based heuristic for solving planning problem," Machine Learning and Cybernetics, 2008 International Conference, Julio 12-15, 2008. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4620775&isnumber=4620699

- Xin Sui; Ho-Fung Leung, "An Adaptive Bidding Strategy in Multi-round Combinatorial Auctions for Resource Allocation," Tools with Artificial Intelligence, 2008. ICTAI '08. 20th IEEE International Conference, Vol 2, pag. 423-430, 3-5 Noviembre, 2008. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4669804&isnumber=4669739

- Xin Yuan; Xingming Liu, "Heuristic algorithms for multi-constrained quality of service routing," INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE , 2001. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=916275&isnumber=19794

- Yang Li; Wei-Xiang Gu; Ming-Hao Yin; Yuan Wasng, "Planning system based on heuristic," Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on , Vol 3, pag. 1385-1390, Agosto 18-21, 2005. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1527160&isnumber=32625

- Zitzler, E.; Thiele, L.; Laumanns, M.; Fonseca, C.: da Fonseca, V. “Performance assessment of multiobjective optimizers: an analysis and review.” IEEE Transactions on Evolutionary Computation, Vol 7, No 2, pag. 117–132, 2003.